



Chapter 5 Convolutional Codes and Trellis Coded Modulation

- 5.1 Encoder Structure and Trellis Representation
- 5.2 Systematic Convolutional Codes
- 5.3 Viterbi Decoding
- 5.4 Soft-Decision Viterbi Decoding
- 5.5 BCJR Decoding
- 5.6 Trellis Coded Modulation



§ 5.1 Encoder Structure and Trellis Representation

- Introduction
 - Encoder: contains memory (order m : m memory units);
 - Output: encoder output at time unit t depends on the input and the memory units status at time unit t ;
 - By increasing the memory order m , one can increase the convolutional code's minimum distance (d_{\min}) and achieve low bit error rate performance (P_b);
 - Decoding Methods:
 - Viterbi algorithm [1]: Maximum Likelihood (ML) decoding algorithm;
 - Bahl, Cocke, Jelinek, and Raviv (BCJR) [2] algorithm: Maximum A Posteriori Probability (MAP) decoding algorithm, used for iterative decoding process, e.g. turbo decoding.

[1] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," IEEE Trans. Inform. Theory, IT-13, 260-269, April, 1967.

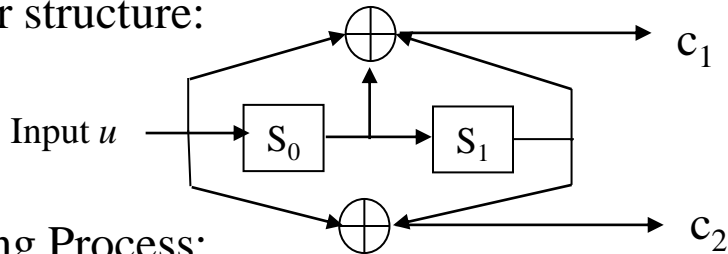
[2] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," IEEE Trans, Inform. Theory, IT-20; 284-287, March, 1974.



§ 5.1 Encoder Structure and Trellis Representation

- The $(7, 5)_8$ conv. code

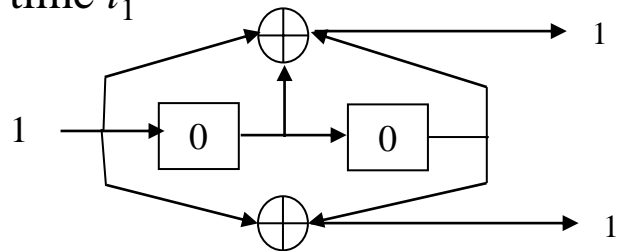
• Encoder structure:



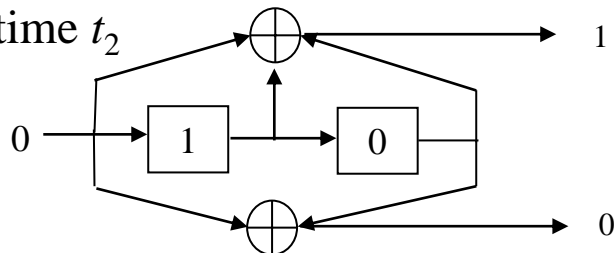
• Encoding Process:

(Initialized state $S_0S_1 = 00$)

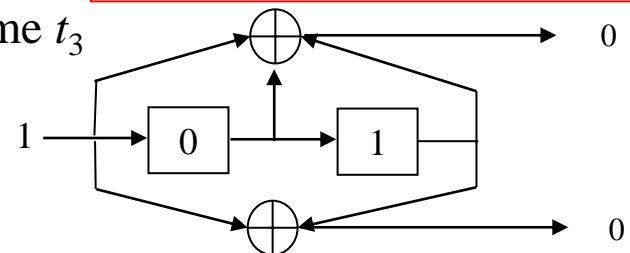
At time t_1



At time t_2



At time t_3



Code rate: $1/2$;

Memory: $m = 2$;

Constraint length: $m + 1 = 3$;

Encoding:

$$c_1 = u \oplus S_0 \oplus S_1;$$

$$c_2 = u \oplus S_1;$$

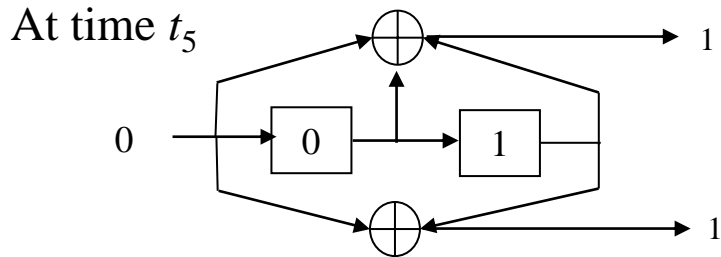
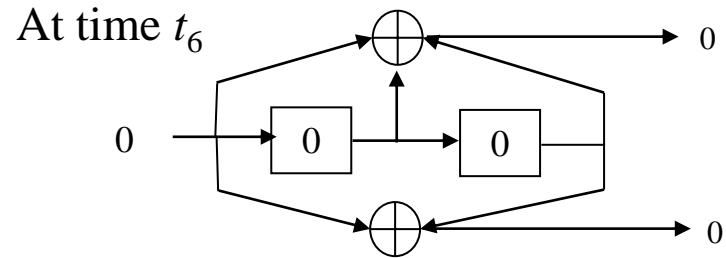
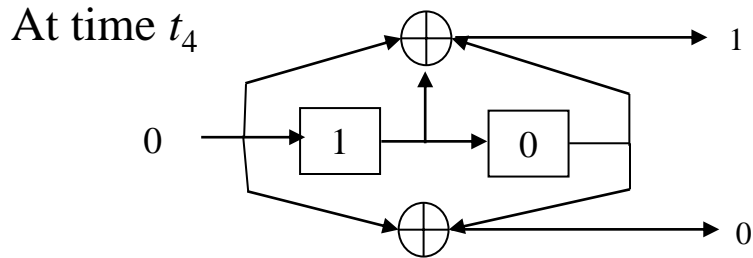
Registers update:

$$S_1' = S_0.$$

$$S_0' = u.$$



§ 5.1 Encoder Structure and Trellis Representation



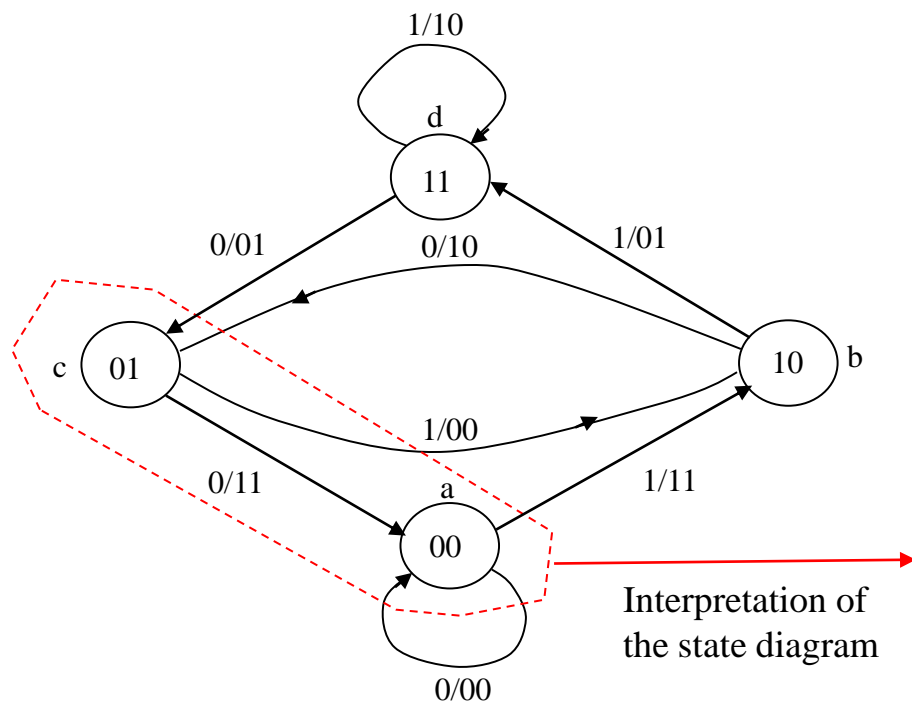
Input sequence $[u_1 u_2 u_3 u_4 u_5 u_6] = [1 0 1 0 0 0]$

Output sequence $[c_1^1 c_1^2 c_2^1 c_2^2 c_3^1 c_3^2 c_4^1 c_4^2 c_5^1 c_5^2 c_6^1 c_6^2] = [11 10 00 10 11 00]$



§ 5.1 Encoder Structure and Trellis Representation

A state transition diagram of the $(7, 5)_8$ conv. code



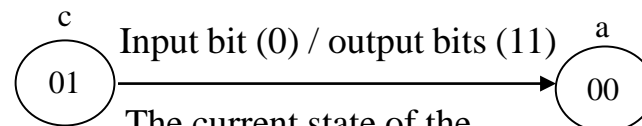
State definition ($S_0 S_1$)

a = 00

b = 10

c = 01

d = 11



Interpretation of the state diagram

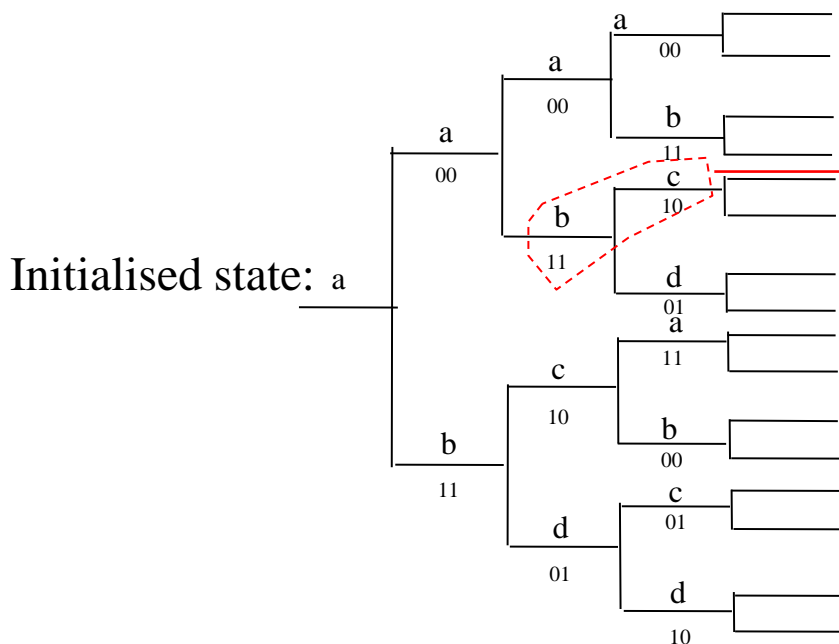
The current state of the encoder is c. If the input bit is 0, it will output 11 and the next state of the encoder is a.



§ 5.1 Encoder Structure and Trellis Representation

Tree Representation of the $(7, 5)_8$ conv. code

Time unit: 1 2 3 4 →



Tree diagram interpretation:

The current state of the encoder is **b**. If the input bit is **0**, the output will be **10**, and the next state of the encoder is **c**.

↑ Input bit as 0 State after transition
 ↓ Input bit as 1 Output from transition

Example 5.1 Determine the codeword that corresponds to message [0 1 1 0 1]

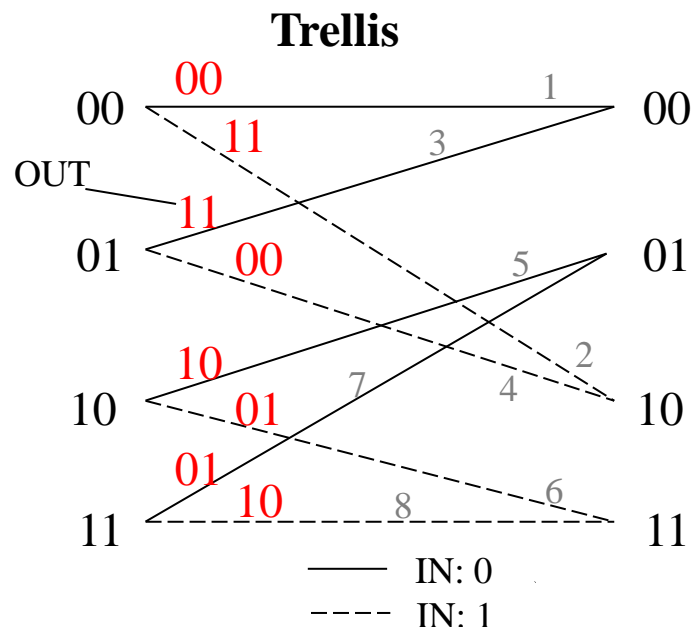


§ 5.1 Encoder Structure and Trellis Representation

Trellis of the $(7, 5)_8$ conv. code

State Table

IN	Current State	Next State	Out	ID
0	00	00	00	1
1	00	10	11	2
0	01	00	11	3
1	01	10	00	4
0	10	01	10	5
1	10	11	01	6
0	11	01	01	7
1	11	11	10	8



Remark: A trellis tells the state transition and IN/OUT relationship. It can be used to yield a convolutional codeword of a sequential input.

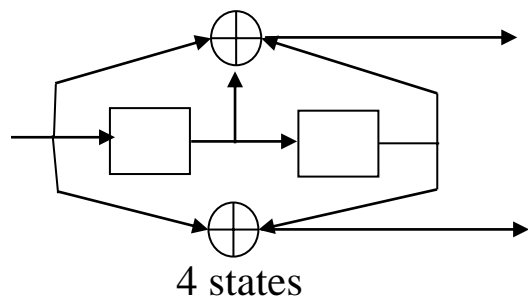
Example 5.2 Use the above trellis to determine the codeword that corresponds to message $[0\ 1\ 1\ 0\ 1]$.



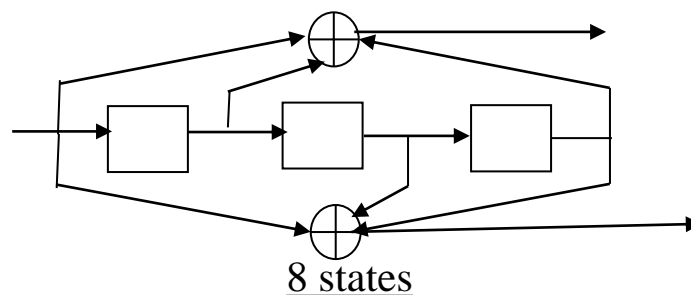
§ 5.1 Encoder Structure and Trellis Representation

Some practical conv. codes

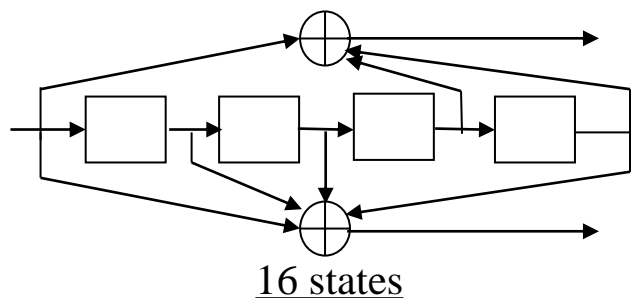
$(7, 5)_8$ conv. code



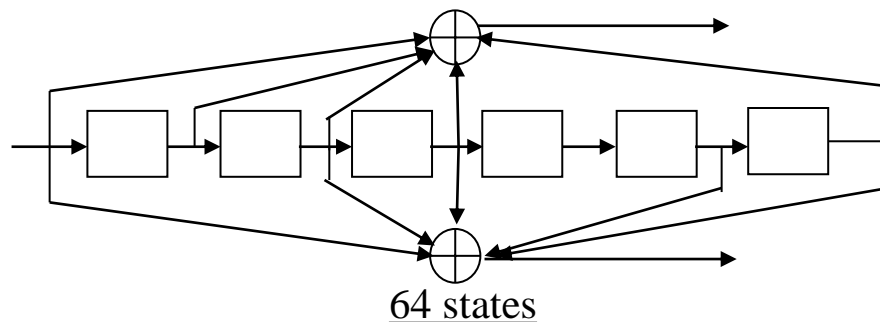
$(15, 13)_8$ conv. code



$(23, 35)_8$ conv. code



$(171, 133)_8$ conv. code



Remark: A convolutional code's error-correction capability improves by increasing the number of the encoder states.

Remark: m trailing bits are needed to force the encoder back to the all-zero state. For the above nonsystematic conv. codes, m 0s are needed.



§ 5.1 Encoder Structure and Trellis Representation

The encoder structure can also be represented by generator sequences or transfer functions.

Example 5.3

- The $(7,5)_8$ conv. code can also be written as:

A rate $\frac{1}{2}$ conv. code with generator sequences

$$g^{(1)} = [1 \ 1 \ 1], \quad g^{(2)} = [1 \ 0 \ 1]$$

A rate $\frac{1}{2}$ conv. code with transfer functions

$$g^{(1)}(x) = 1 + x + x^2, \quad g^{(2)}(x) = 1 + x^2$$

- The $(15,13)_8$ conv. code can also be written as:

A rate $\frac{1}{2}$ conv. code with generator sequences:

$$g^{(1)} = [1 \ 1 \ 0 \ 1], \quad g^{(2)} = [1 \ 0 \ 1 \ 1]$$

A rate $\frac{1}{2}$ conv. code with transfer functions:

$$g^{(1)}(x) = 1 + x + x^3, \quad g^{(2)}(x) = 1 + x^2 + x^3$$



§ 5.1 Encoder Structure and Trellis Representation

When the codeword (message) length is finite, conv. code is also a linear block code. That says its encoding can be defined by $\bar{c} = \bar{u} \cdot \mathbf{G}$

Example 5.4 For the (7,5) convolutional code with message $\bar{u} = (u_1, u_2, u_3, u_4, u_5)$, we have

$$u_1 \rightarrow \begin{cases} c_1^1 = 0 \oplus 0 \oplus u_1 \\ c_1^2 = 0 \oplus u_1 \end{cases} \quad u_2 \rightarrow \begin{cases} c_2^1 = 0 \oplus u_1 \oplus u_2 \\ c_2^2 = 0 \oplus u_2 \end{cases} \quad u_3 \rightarrow \begin{cases} c_3^1 = u_1 \oplus u_2 \oplus u_3 \\ c_3^2 = u_1 \oplus u_3 \end{cases}$$

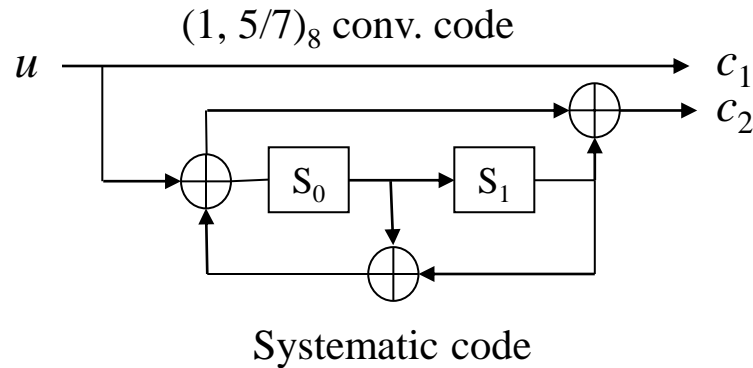
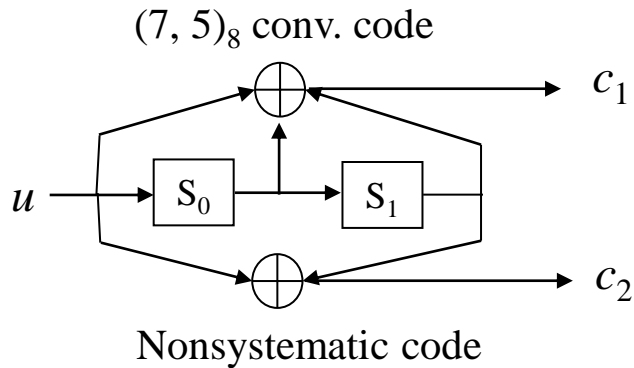
$$u_4 \rightarrow \begin{cases} c_4^1 = u_2 \oplus u_3 \oplus u_4 \\ c_4^2 = u_2 \oplus u_4 \end{cases} \quad u_5 \rightarrow \begin{cases} c_5^1 = u_3 \oplus u_4 \oplus u_5 \\ c_5^2 = u_3 \oplus u_5 \end{cases}$$

$$\mathbf{G} = \begin{matrix} & c_1^1 & c_1^2 & c_2^1 & c_2^2 & c_3^1 & c_3^2 & c_4^1 & c_4^2 & c_5^1 & c_5^2 \\ \begin{matrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{matrix} & \left[\begin{array}{cccccccccc} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array} \right] \end{matrix}$$



§ 5.2 Systematic Convolutional Codes

- The $(7, 5)_8$ conv. code's systematic counterpart is:



Encoding and Registers' updating rules:
[$S_0 S_1$] are initialized as [0 0];
 $c_1 = u$; (systematic feature) feedback = $S_0 \oplus S_1$;
 $c_2 = u \oplus \text{feedback} \oplus S_1$; $S_1' = S_0$; $S_0' = u \oplus \text{feedback}$;

Remark: Systematic encoding structure is often used to constitute Turbo codes (Chapter 6).



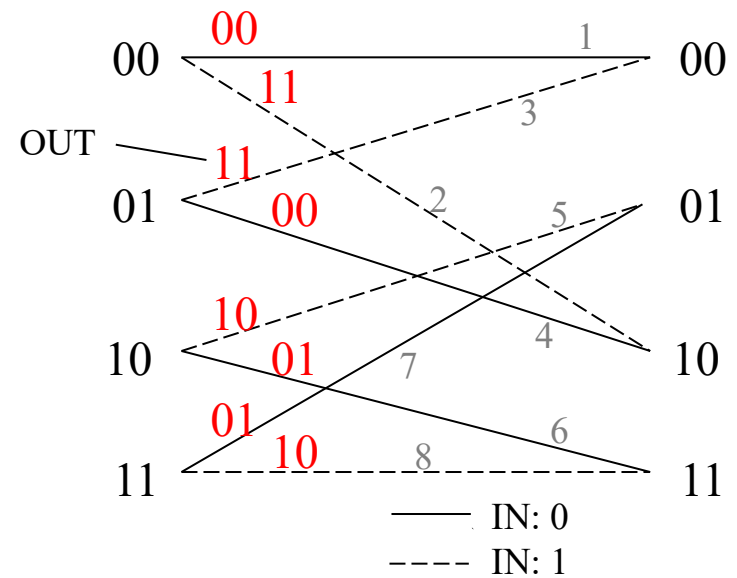
§ 5.2 Systematic Convolutional Codes

For the $(1, 5/7)_8$ conv. code

State Table

IN	Current State	Next State	Out	ID
0	00	00	00	1
1	00	10	11	2
0	01	10	00	3
1	01	00	11	4
0	10	11	01	5
1	10	01	10	6
0	11	01	01	7
1	11	11	10	8

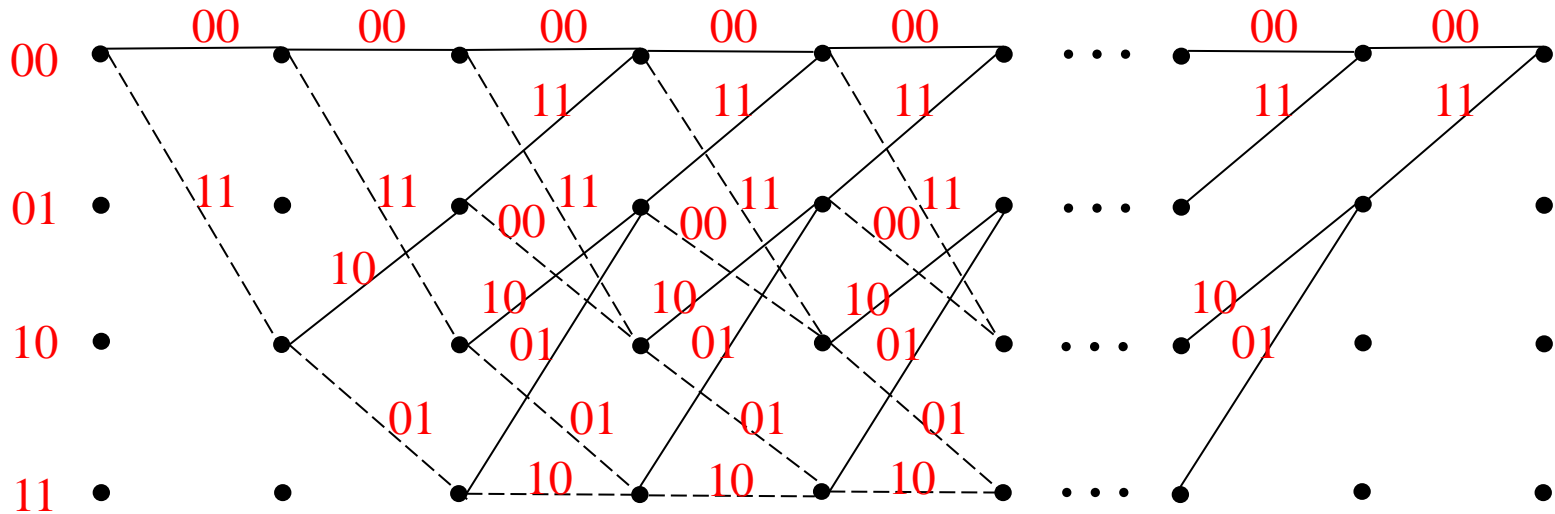
Trellis





§ 5.3 Viterbi Decoding

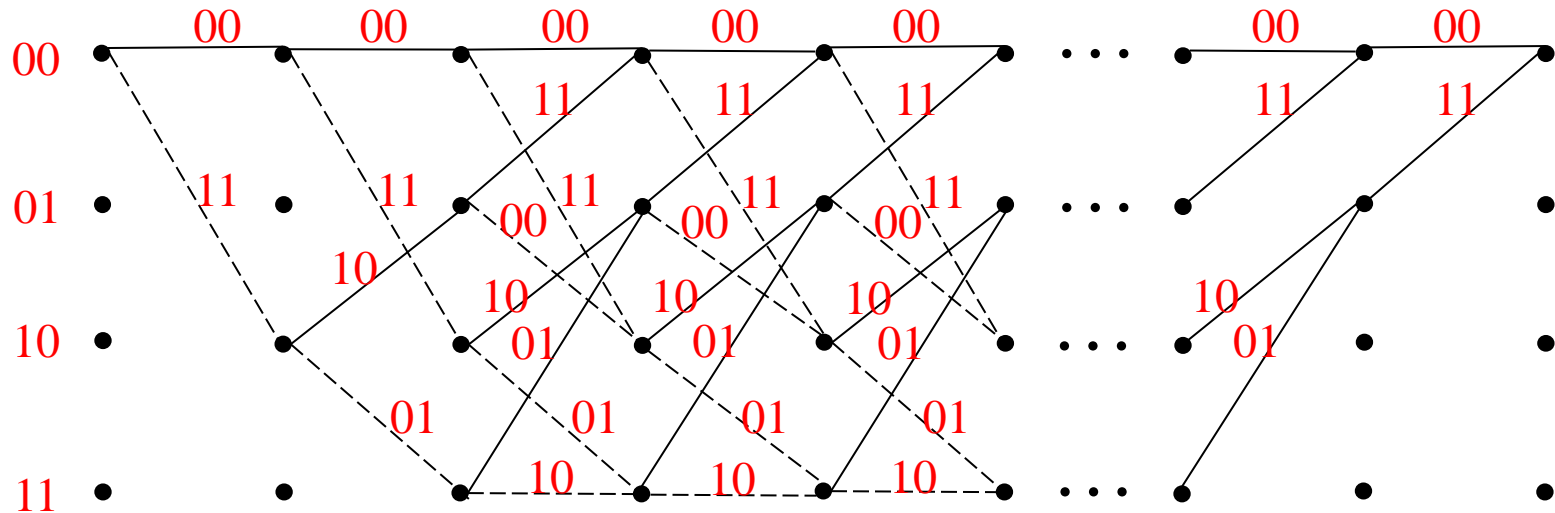
Let us extend the trellis of the $(7, 5)_8$ conv. code as if there is a sequential input.



- Such an extension results in a **Viterbi trellis**
- A path in the Viterbi trellis represents a convolutional codeword that corresponds to a sequential input (message).



§ 5.3 Viterbi Decoding



- Decoding motivation: Given a received word \bar{r} , find the mostly likely codeword \hat{c} such that the Hamming distance $d(\bar{r}, \hat{c})$ is minimized.
- Since \hat{c} corresponds to a path in the Viterbi trellis, tracing back the path of \hat{c} enables us to find out the message.
- Branch metrics: Hamming distance between a transition branch's output and the corresponding received symbol (or bits).
- Path metrics: Accumulated Hamming distance of the previous branch metrics.



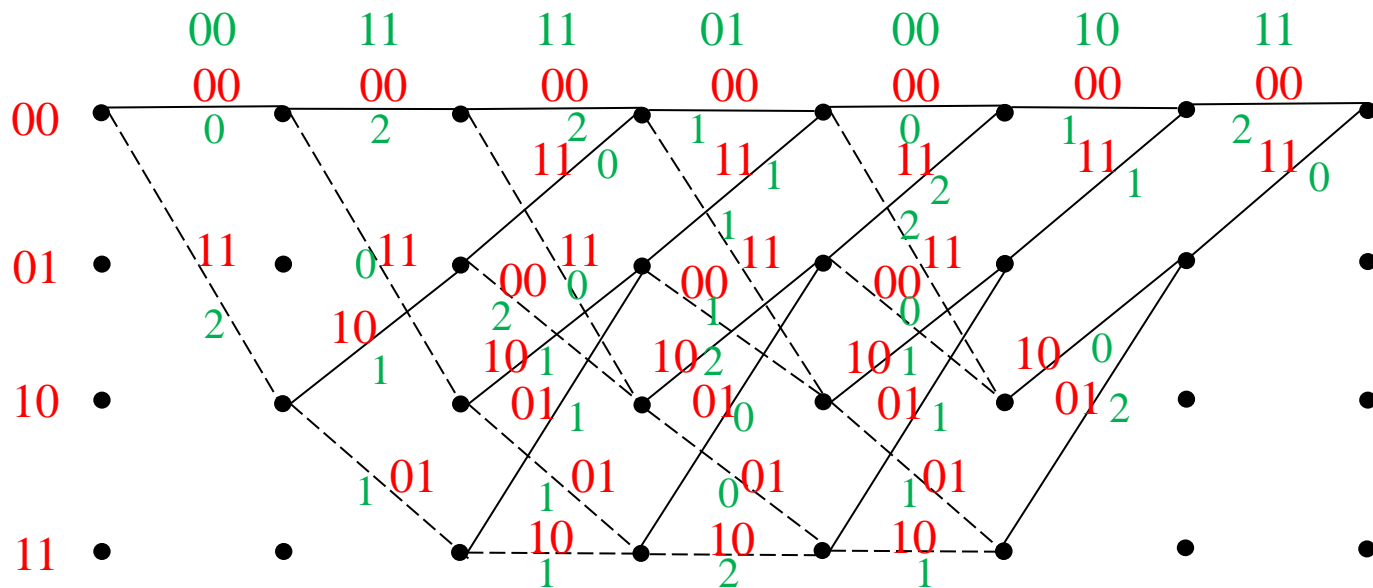
§ 5.3 Viterbi Decoding

Example 5.5 Given the $(7, 5)_8$ conv. code as in *Examples 5.1 - 5.3*. The transmitted codeword is $\bar{c} = [00\ 11\ 01\ 01\ 00\ 10\ 11]$ (codeword produced by adding tailing bits). After channel, the received word is

$$\bar{r} = [00\ 11\ \boxed{11}\ 01\ 00\ 10\ 11]$$

Try to use the following Viterbi trellis to decode it.

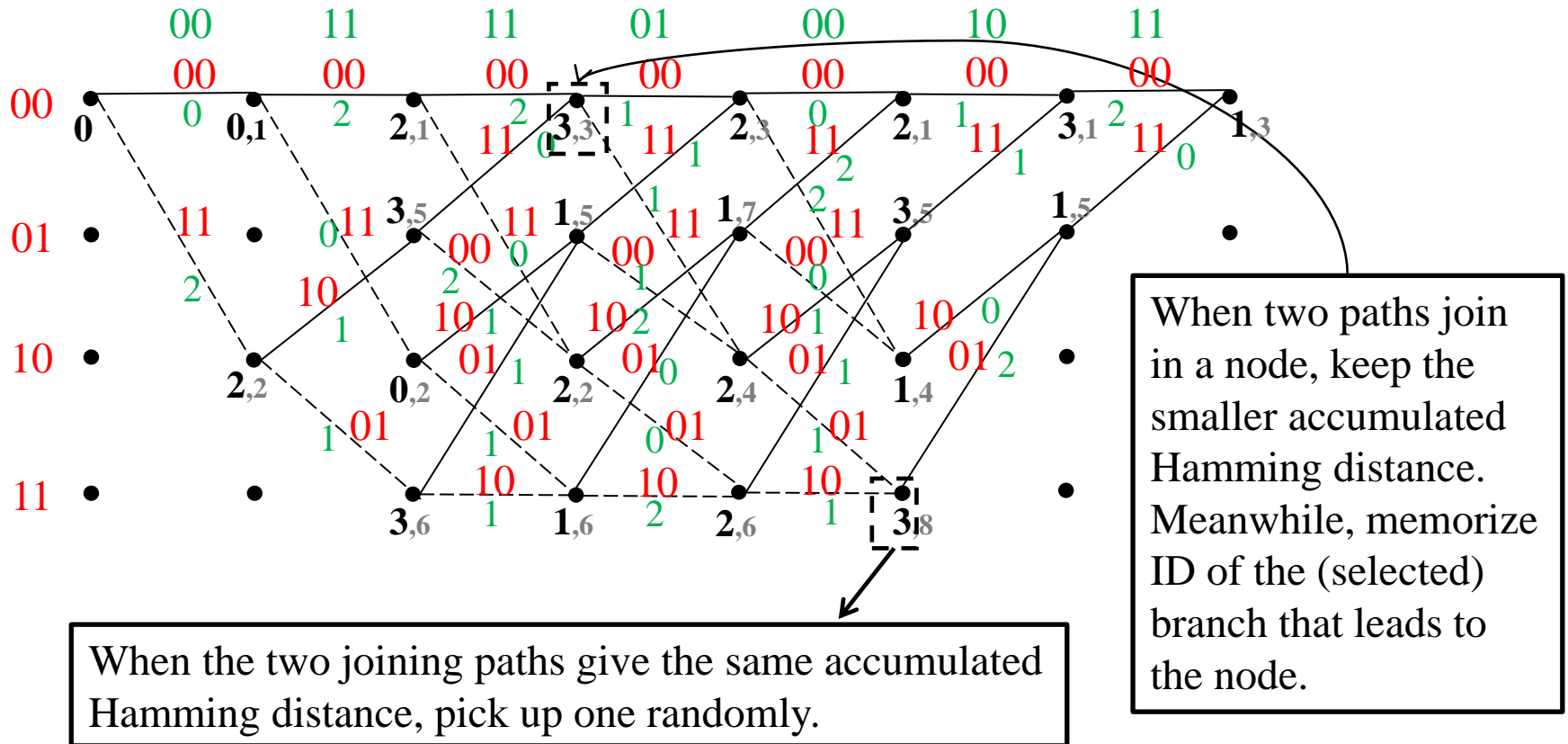
Step 1: Calculate all the branch metrics.





§ 5.3 Viterbi Decoding

Step 2: Calculate the path metrics and memorize the path IDs.



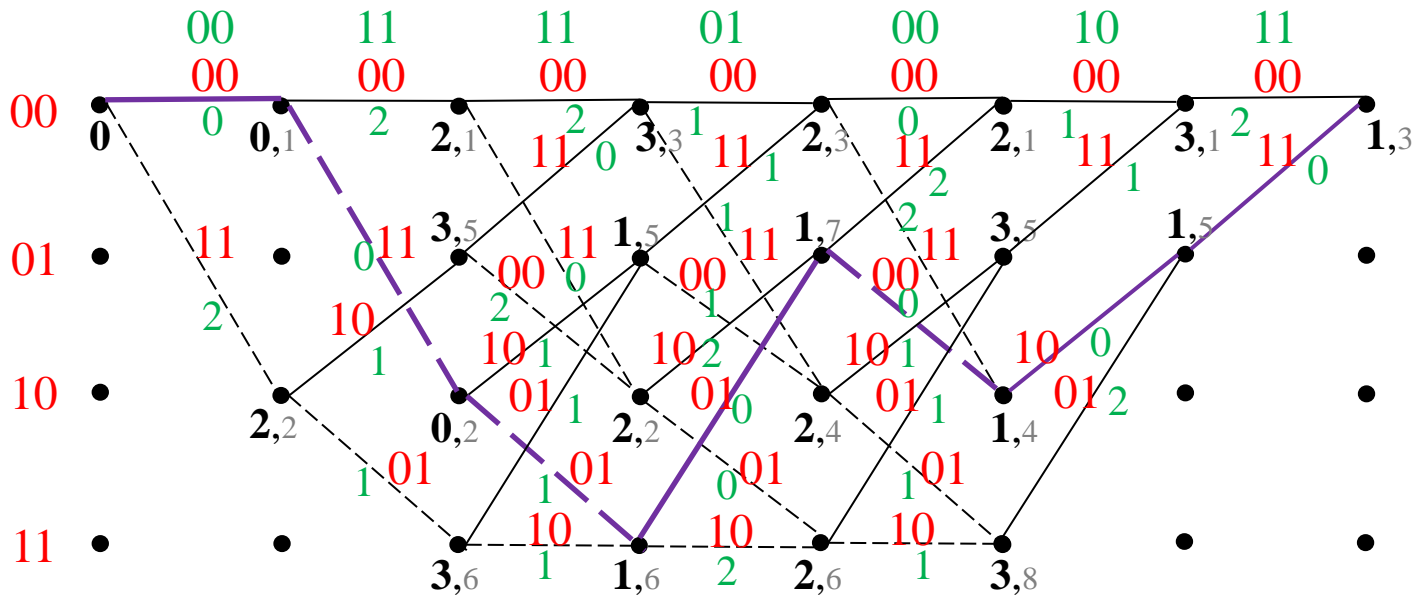


§ 5.3 Viterbi Decoding

Step 3: Pick up the minimal path metric and trace back to determine the message.

Tracing rules: (1) Trellis connection;

(2) The tracing route should match the trellis transition ID.



Decoding output: 0 1 1 0 1 0 0



§ 5.3 Viterbi Decoding

Metrics of the Viterbi Decoding Process

Branch Metrics Table

ID	Branch Metric						
1	0	2	2	1	0	1	2
2	2	0	0	1	2	∞	∞
3	∞	∞	0	1	2	1	0
4	∞	∞	2	1	0	∞	∞
5	∞	1	1	2	1	0	∞
6	∞	1	1	0	1	∞	∞
7	∞	∞	1	0	1	2	∞
8	∞	∞	1	2	1	∞	∞

Path Metrics Table

00	0	0	2	3	2	2	3	1
01	∞	∞	3	1	1	3	1	∞
10	∞	2	0	2	2	1	∞	∞
11	∞	∞	3	1	2	3	∞	∞

Trellis Transition ID Table

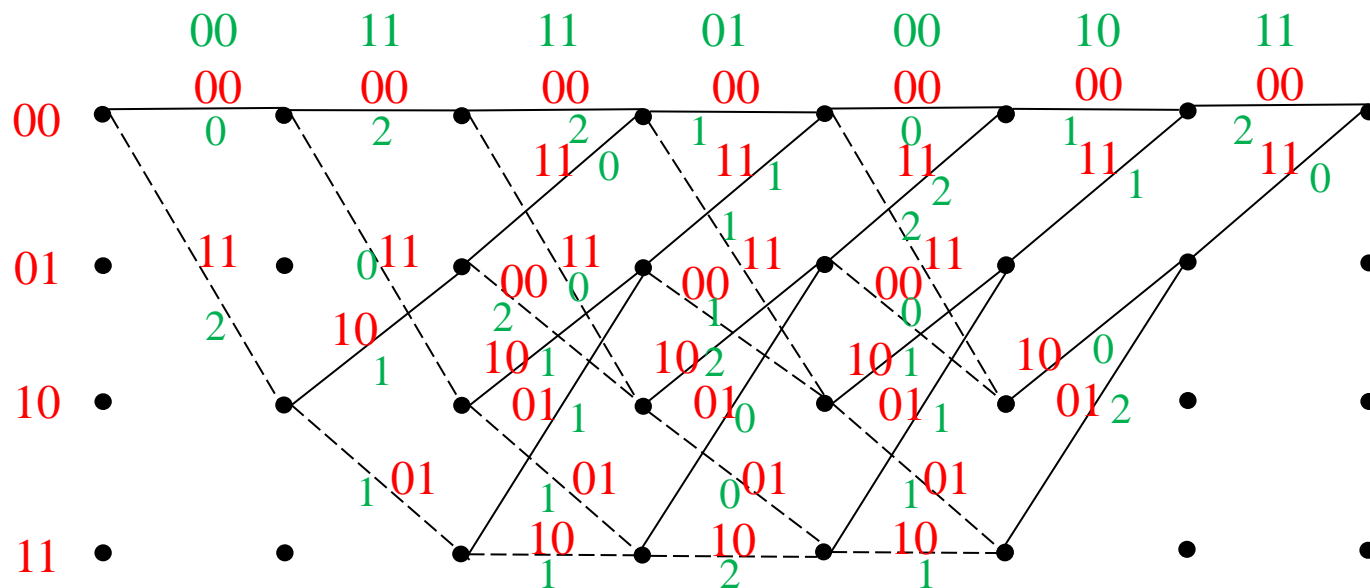
00	1	1	3	3	1	1	3
01	×	5	5	7	5	5	×
10	2	2	2	4	4	×	×
11	×	6	6	6	8	×	×

Remark: With tailing bits, the backward trace always starts from the all-zero state.



§ 5.3 Viterbi Decoding

Free distance of a convolutional code



- A conv. code's performance is determined by its free distance.
- Free distance

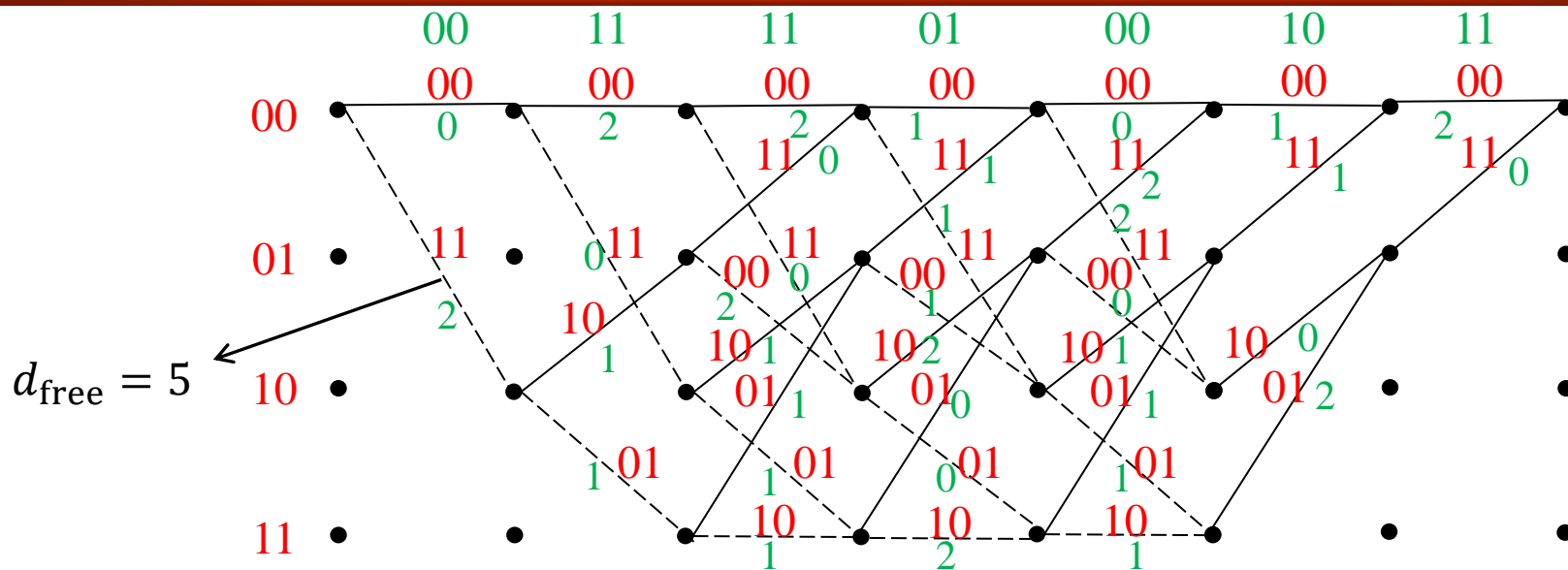
$$d_{\text{free}} = \min\{d_{\text{Ham}}(\bar{c}_1, \bar{c}_2), \bar{c}_1 \neq \bar{c}_2\}$$

- With knowing $\mathbf{0} = [0\ 0\ 0\ \dots\ 0]$ is also a convolutional codeword

$$d_{\text{free}} = \min\{\text{weight}(\bar{c}), \bar{c} \neq \mathbf{0}\}.$$



§ 5.3 Viterbi Decoding



Hence, it is the minimum weight of all finite length paths in the Viterbi trellis that diverge from and emerge with the all zero state.

- The tailing bits are needed to ensure a greater free distance for the code.
- Convolutional code with a large number of states will have a great d_{free} , and hence stronger error-correction capability



§ 5.3 Viterbi Decoding

Remark: Convolutional code is more competent in correcting spread errors, but not burst errors.

E.g., with $\bar{r}_1 = [0 \ 1 \ e \ 1 \ e \ 1 \ 0 \ 1 \ 0 \ 0 \ e \ 1]$
and $\bar{r}_2 = [0 \ 1 \ 0 \ e \ e \ e \ 0 \ 1 \ 0 \ 0 \ 1 \ 1]$

error
↙

Viterbi algorithm is more competent in correcting received vector \bar{r}_1 .



§ 5.4 Soft-Decision Viterbi Decoding

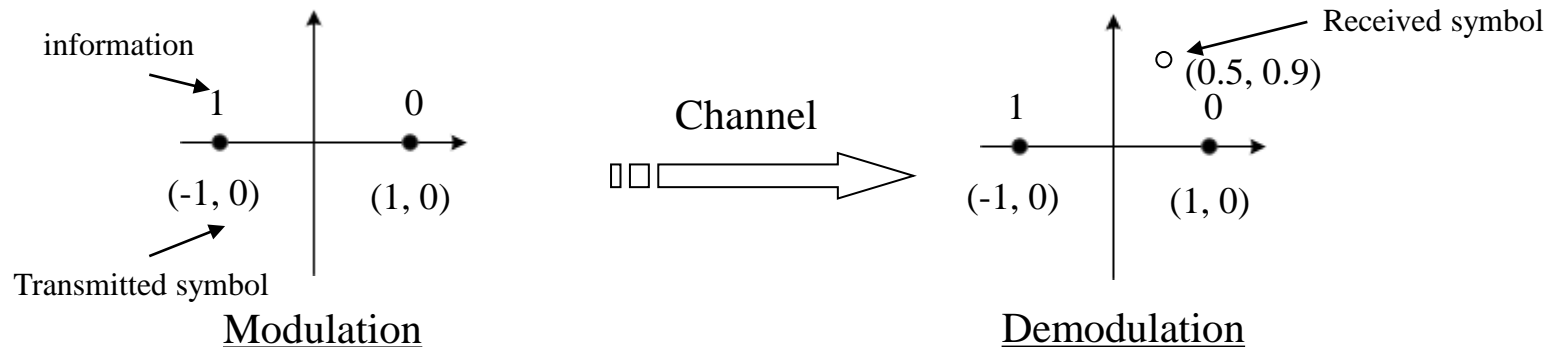
➤ Soft-decision Viterbi decoding

- While we are performing the hard-decision Viterbi decoding, we have the scenario that two joining paths yield the same accumulated Hamming distance. This would cause decoding ‘ambiguity’ and performance penalty;
- Such a performance loss can be compensated by utilizing soft-decision decoding, e.g., soft-decision Viterbi decoding

➤ Modulation and Demodulation

- Modulation: mapping the coded symbol into a transmitted symbol;
- Demodulation: determining the codeword symbol with a received symbol;

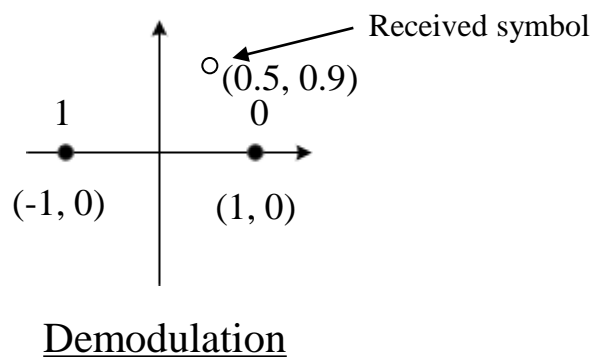
BPSK





§ 5.4 Soft-Decision Viterbi Decoding

- Modulation and Demodulation (e.g., BPSK)

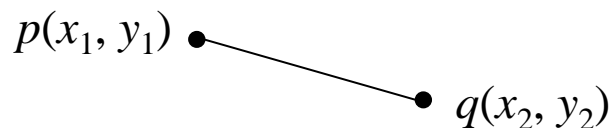


Hard-decision: the information bit is 0. The Hamming distance becomes the Viterbi decoding metrics;

Soft-decision: the information bit has Pr. of 0.7 being 0 and Pr. of 0.3 being 1. The *Euclidean distance (or probability)* becomes the Viterbi decoding metrics;

- Euclidean Distance

Definition: The Euclidean distance between points p and q is the length of the line segment connecting them.



$$d_{Eud} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



§ 5.4 Soft-Decision Viterbi Decoding

Example 5.6. Given the $(7, 5)_8$ conv. code as in *Examples 5.5*.

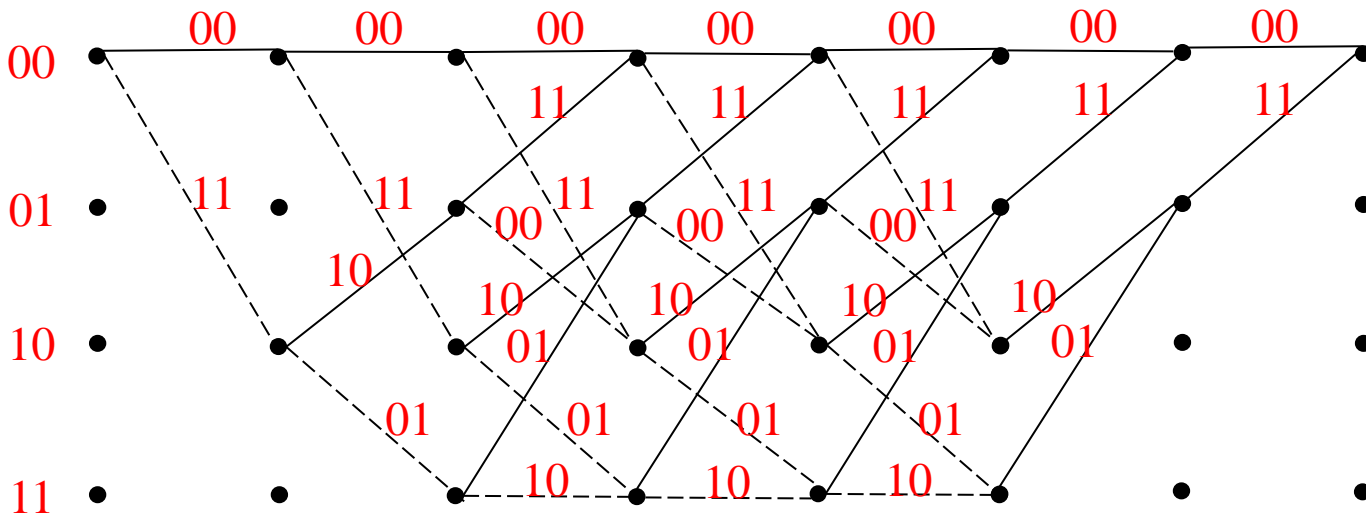
The transmitted codeword is $\bar{c} = [00\ 11\ 01\ 01\ 00\ 10\ 11]$

After BPSK modulation, the transmitted symbols are:

$(1, 0), (1, 0), (-1, 0), (-1, 0), (1, 0), (-1, 0), (1, 0), (-1, 0), (1, 0), (1, 0), (-1, 0), (1, 0), (-1, 0), (-1, 0)$.

After the channel, the received symbols are:

$(0.8, 0.2), (1.2, -0.4), (-1.3, 0.3), (-0.9, -0.1), (-0.5, 0.4), (-1.0, 0.1), (1.1, 0.4), (-0.7, -0.2),$
 $(1.2, 0.2), (0.9, 0.3), (-0.9, -0.2), (1, 0.2), (-1.1, 0), (-0.8, 0.1)$.



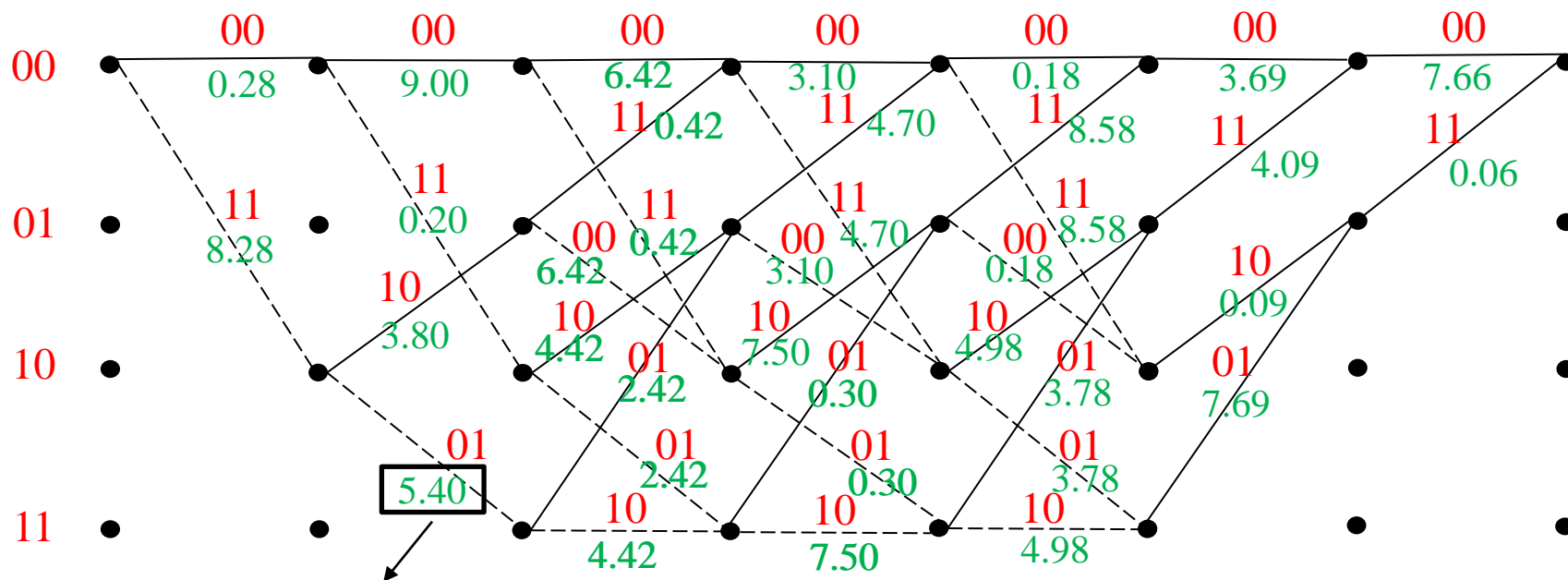


§ 5.4 Soft-Decision Viterbi Decoding

Step 1: Calculate all the branch metrics.

0 → (1, 0)
1 → (-1, 0)

(0.8, 0.2) (-1.3, 0.3) (-0.5, 0.4) (1.1, 0.4) (1.2, 0.2) (-0.9, -0.2) (-1.1, 0)
 (1.2, -0.4) (-0.9, -0.1) (-1.0, 0.1) (-0.7, -0.2) (0.9, 0.3) (1, 0.2) (-0.8, 0.1)



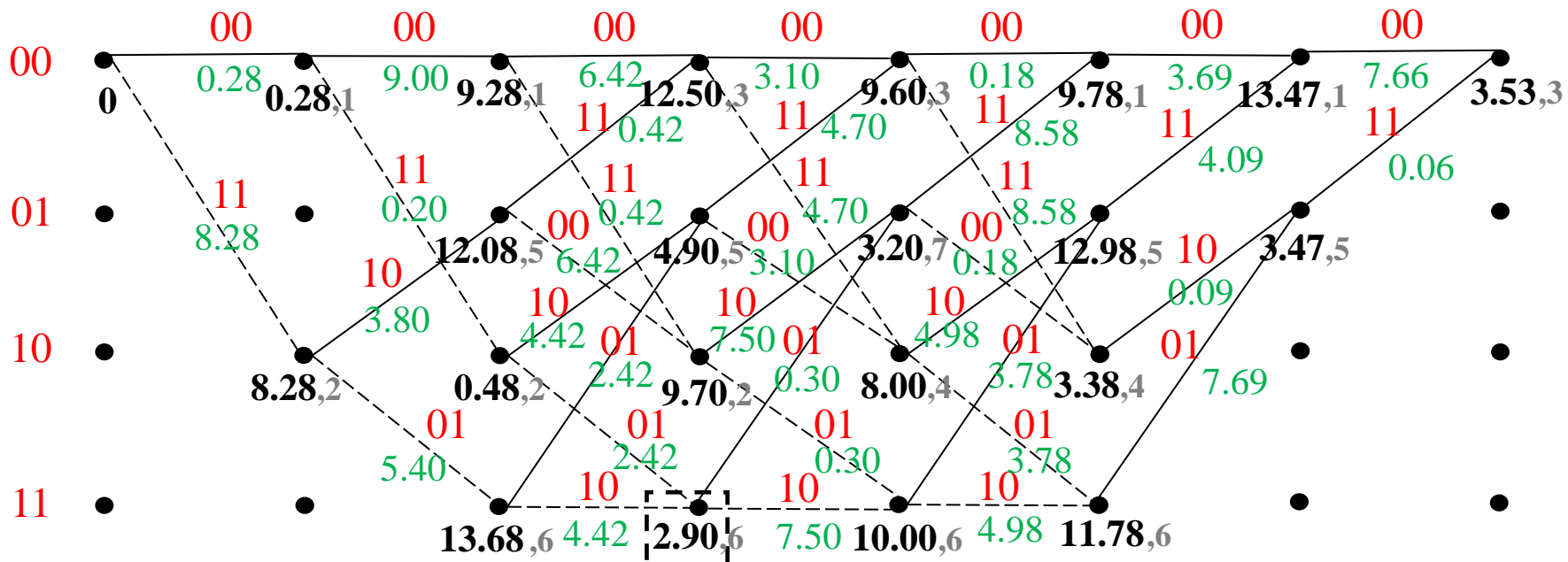
Branch metric: $(-1.3 - 1)^2 + (0.3 - 0)^2 + (-0.9 + 1)^2 + (-0.1 - 0)^2 = 5.40$



§ 5.4 Soft-Decision Viterbi Decoding

Step 2: Calculate the path metrics and memorize the path IDs.

(0.8, 0.2) (-1.3, 0.3) (-0.5, 0.4) (1.1, 0.4) (1.2, 0.2) (-0.9, -0.2) (-1.1, 0)
 (1.2, -0.4) (-0.9, -0.1) (-1.0, 0.1) (-0.7, -0.2) (0.9, 0.3) (1, 0.2) (-0.8, 0.1)



When two paths join in a node, keep the smaller accumulated squared Euclidean distance.

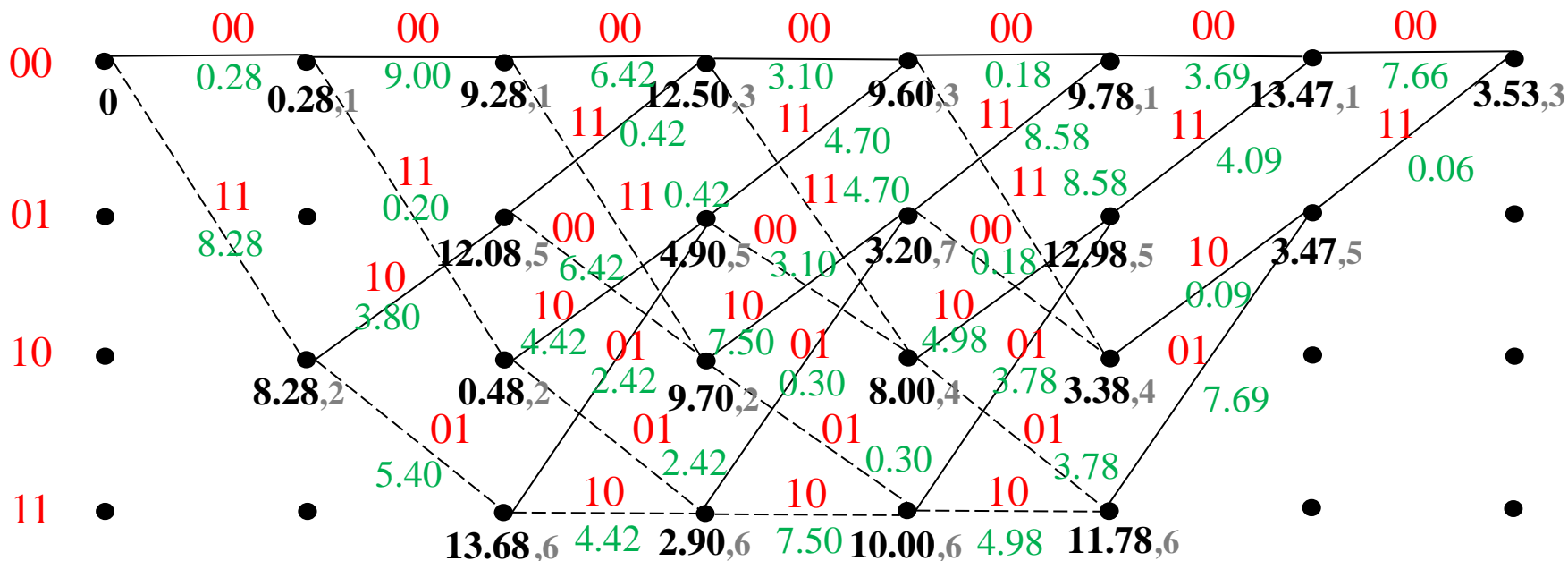


§ 5.4 Soft-Decision Viterbi Decoding

Step 3: Pick up the minimal path metric and trace back to determine the message.

Tracing rules: The same as hard-decision Viterbi decoding algorithm.

(0.8, 0.2) (-1.3, 0.3) (-0.5, 0.4) (1.1, 0.4) (1.2, 0.2) (-0.9, -0.2) (-1.1, 0)
 (1.2, -0.4) (-0.9, -0.1) (-1.0, 0.1) (-0.7, -0.2) (0.9, 0.3) (1, 0.2) (-0.8, 0.1)



Decoding output: 0 1 1 0 1 0 0



§ 5.4 Soft-Decision Viterbi Decoding

Metrics of the Soft-Decision Viterbi Decoding Process

Path Metrics Table

00	0	0.28	0.98	12.50	9.60	9.78	13.47	3.53
01	∞	∞	12.08	4.90	3.20	12.98	3.47	∞
10	∞	8.28	0.48	9.70	8.00	3.38	∞	∞
11	∞	∞	13.68	2.90	10.00	11.78	∞	∞

Branch Metrics Table

ID	Branch Metric						
1	0.28	9.00	6.42	3.10	0.18	3.69	7.66
2	8.28	0.20	0.42	4.70	8.58	∞	∞
3	∞	∞	0.42	4.70	8.58	4.09	0.06
4	∞	∞	6.42	3.10	0.18	∞	∞
5	∞	3.80	4.42	7.50	4.98	0.09	∞
6	∞	5.40	2.42	0.30	3.78	∞	∞
7	∞	∞	2.42	0.30	3.78	7.69	∞
8	∞	∞	4.42	7.50	4.98	∞	∞

Trellis Transition ID Table

00	1	1	3	3	1	1	3
01	×	5	5	7	5	5	×
10	2	2	2	4	4	×	×
11	×	6	6	6	6	×	×



§ 5.5 BCJR Decoding

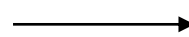
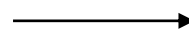
- Hard-decision Viterbi Decoding : a Hard-In-Hard-Out (HIHO) decoding.
Soft-decision Viterbi Decoding : a Soft-In-Hard-Out (SIHO) decoding.
BCJR Decoding: a Soft-In-Soft-Out (SISO) decoding.
- A Soft-In-Soft-Out (SISO) decoding algorithm that takes probabilities as the input and delivers probabilities as the output.
- With an attempt to deliver both the *a posteriori probabilities* of $P(c_t|y_t)$ and $P(u_{t'}|y_t)$, it is also called the maximum *a posteriori* (MAP) algorithm.

In light of a rate 1/2 code

message symbols: $u_1, u_2, \dots, u_{t'}, \dots$

codeword symbols: $c_1, c_2, \dots, c_t, \dots$

received symbols: $y_1, y_2, \dots, y_t, \dots$



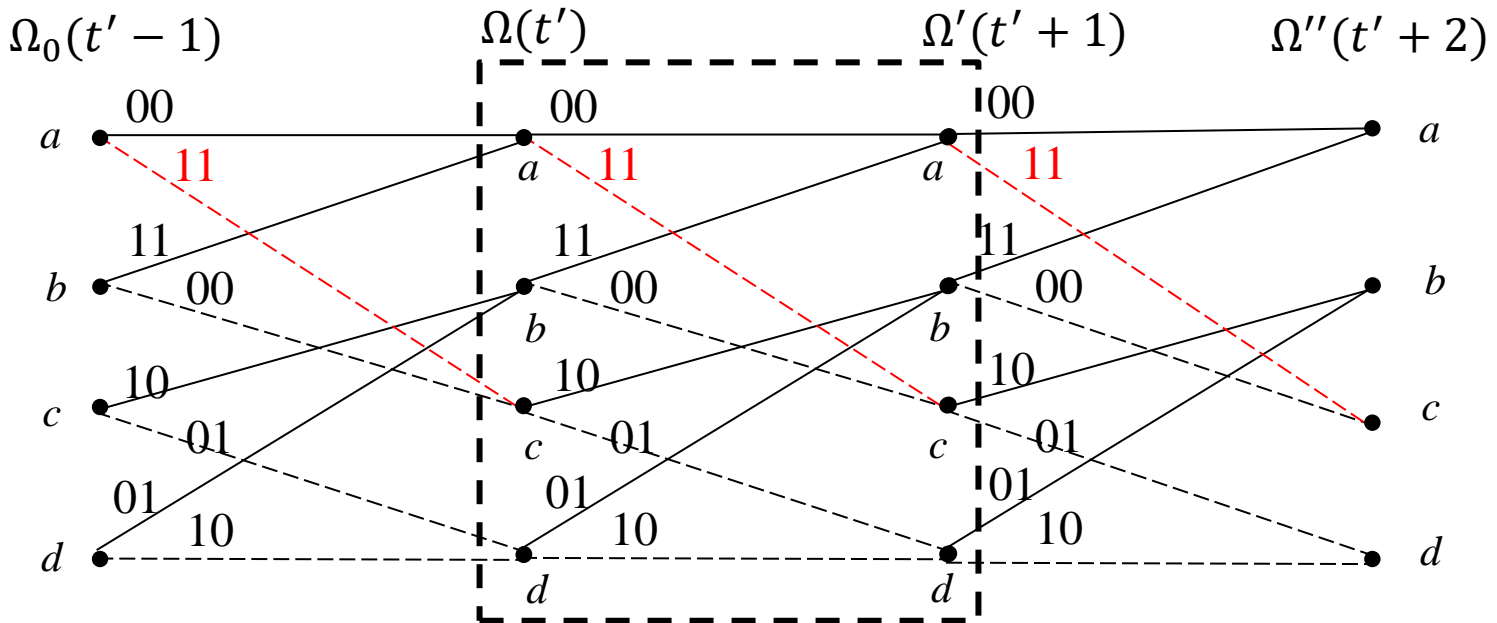
$c_1^1, c_1^2, c_2^1, c_2^2, \dots, c_{t'}^1, c_{t'}^2, \dots$

$y_1^1, y_1^2, y_2^1, y_2^2, \dots, y_{t'}^1, y_{t'}^2, \dots$



§ 5.5 BCJR Decoding

- In a trellis (e.g., trellis of the $(7, 5)_8$ conv. code).

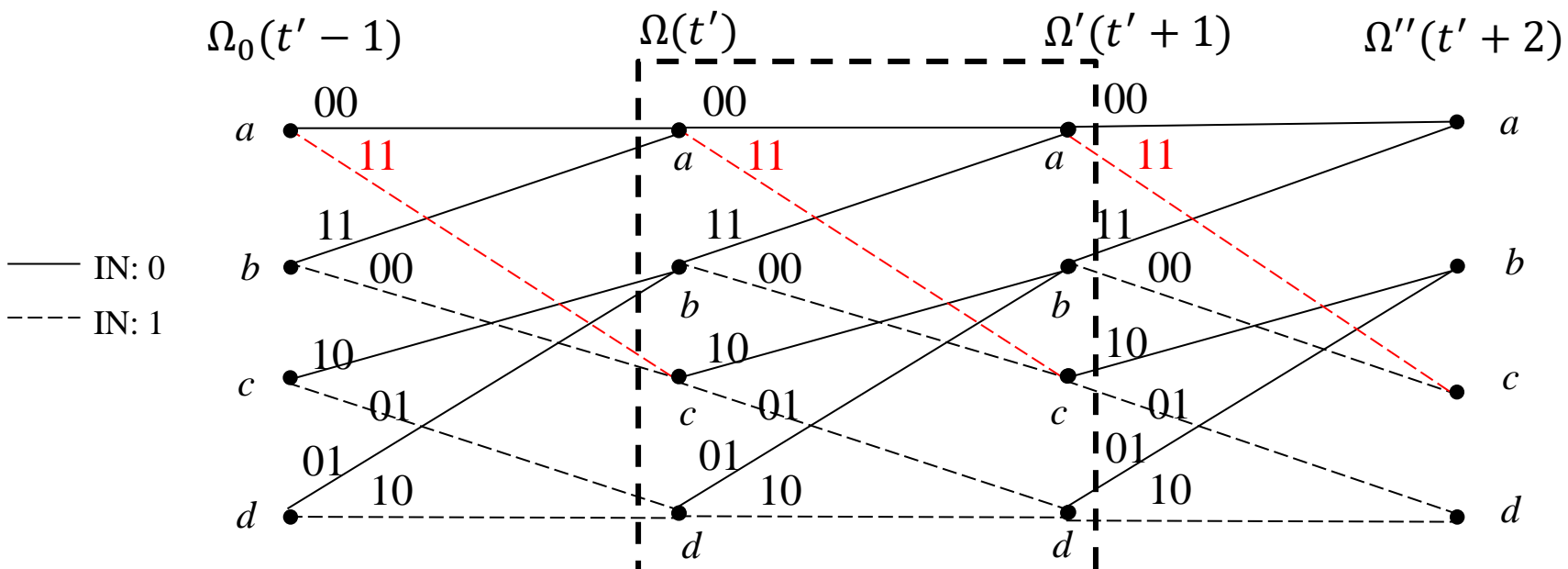


The (IN, OUT, current state, next state) tuple happens as an entity.

—— IN: 0
 ---- IN: 1



§ 5.5 BCJR Decoding



That says at time instant t'

$$\sum \text{Prob} [\text{trellis transition w.r.t. an input } \theta] = \text{Prob} [u_{t'} = \theta], \theta \in \{0,1\}.$$

$$\sum \text{Prob} [\text{trellis transition w.r.t. an output of } \theta] = \text{Prob} [c_{t'}^{1(2)} = \theta], \theta \in \{0,1\}.$$

We seek to determine all the Prob[trellis transition w.r.t. an **input** θ] at time t' to know

$$P(u_{t'} = \theta | y_t)$$

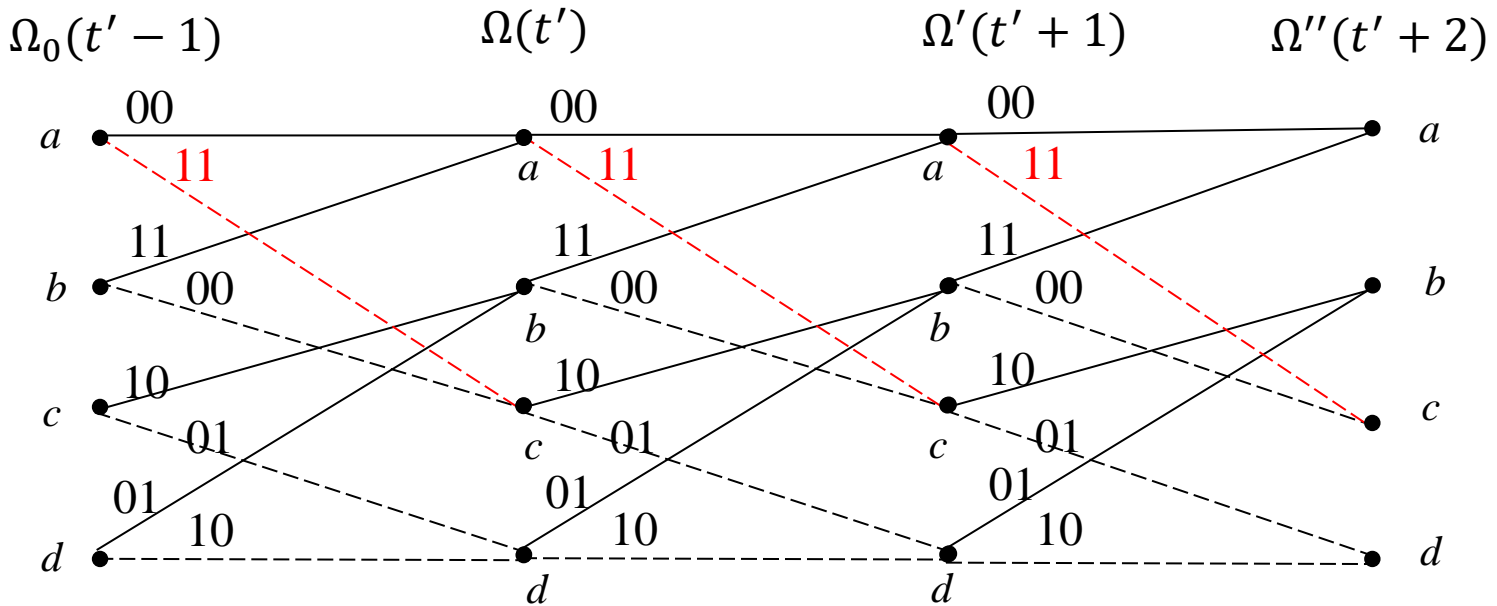
We seek to determine all the Prob[trellis transition w.r.t. an **output** θ] at time t' to know

$$P(c_{t'}^{1(2)} = \theta | y_t)$$



§ 5.5 BCJR Decoding

- Determine the state transition probabilities.



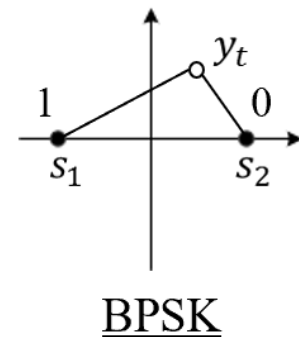
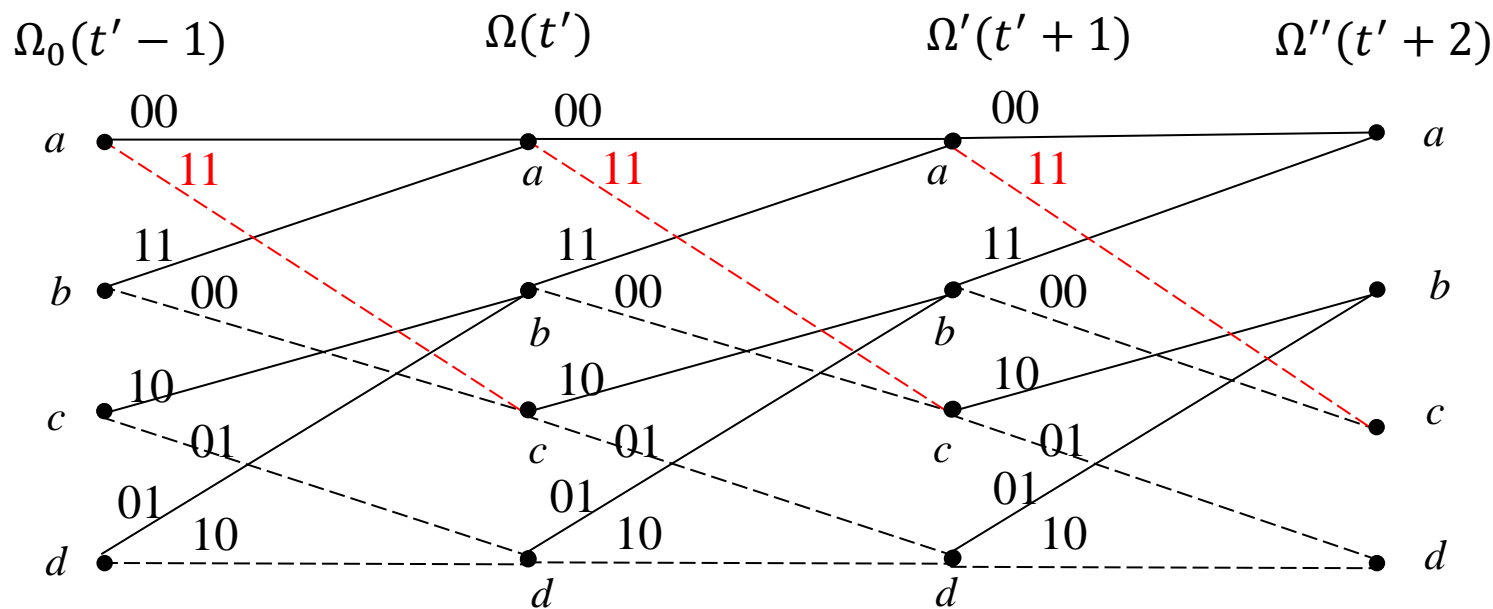
- For a rate half conv. code, $u_{t'} \rightarrow c_{t'}^1, c_{t'}^2$
- Trellis state transition probability: $(\Omega, \Omega') \in \{a, b, c, d\}$

$$\Gamma_{\Omega \rightarrow \Omega'} = P_a(u_{t'}) P_{ch}(c_{t'}^1) P_{ch}(c_{t'}^2)$$



§ 5.5 BCJR Decoding

- Determine the state transition probabilities.



$$\Gamma_{\Omega \rightarrow \Omega'} = P_a(u_{t'}) P_{ch}(c_{t'}^1) P_{ch}(c_{t'}^2)$$

A priori prob. of information bit. E.g. w/o knowledge of $u_{t'}$,
 $P_a(u_{t'} = 0) = P_a(u_{t'} = 1) = 0.5$

Channel observations: E.g. BPSK is used

$$P_{ch}(c_{t'}^1 = 0) = P(y_t | c_{t'}^1 = 0) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{\|y_t - s_2\|^2}{N_0}\right)$$

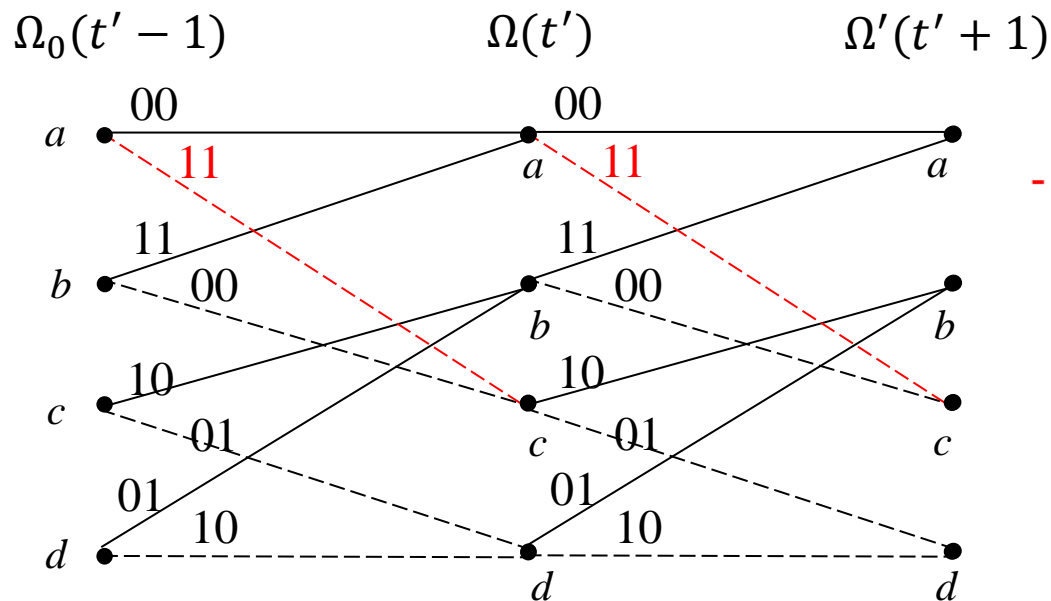
$$P_{ch}(c_{t'}^1 = 1) = P(y_t | c_{t'}^1 = 1) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{\|y_t - s_1\|^2}{N_0}\right)$$

$P_{ch}(c_{t'}^2)$ can be calculated similarly.



§ 5.5 BCJR Decoding

- Determine the probability of each beginning state.



- Probability of beginning a trellis transition ($\Omega \rightarrow \Omega'$) from state Ω (Determined by a forward trace).

$$A_{t'}(\Omega) = N_A \sum_{(\Omega_0, \Omega)} A_{t'-1}(\Omega_0) \cdot \Gamma_{\Omega_0 \rightarrow \Omega},$$

$$t' = 1, 2, \dots, k.$$

N_A : Normalization factor that ensures $A_{t'}(a) + A_{t'}(b) + A_{t'}(c) + A_{t'}(d) = 1$.

- Knowing the Viterbi trellis starts from the all-zero state, we initialize:

$$A_0(a) = 1, \text{ and } A_0(b) = A_0(c) = A_0(d) = 0.$$

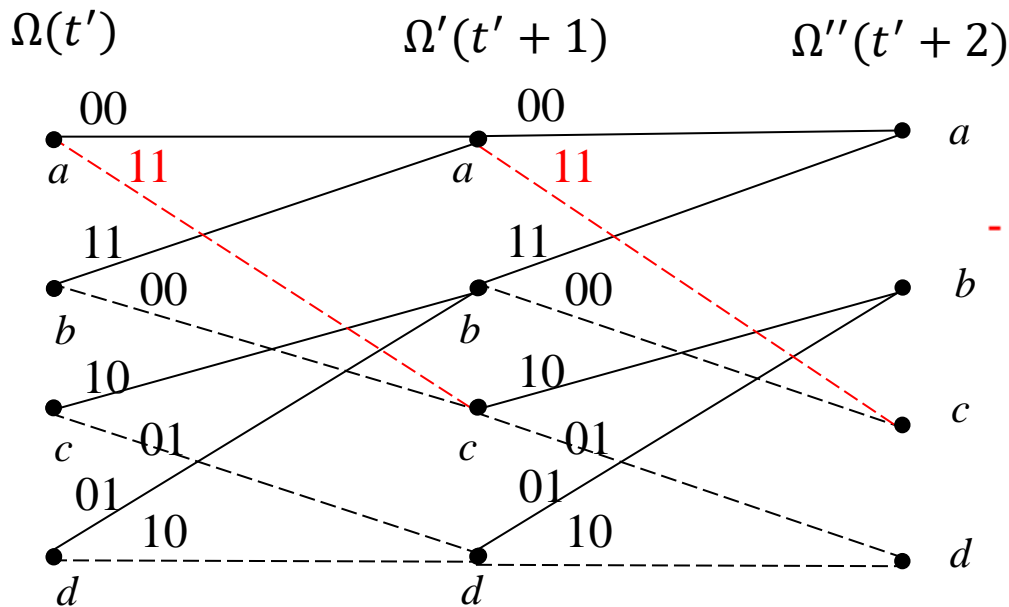
- E.g., in the highlighted trellis transition

$$A_{t'}(a) = A_{t'-1}(a) \cdot \Gamma_{a \rightarrow a} + A_{t'-1}(b) \cdot \Gamma_{b \rightarrow a}.$$



§ 5.5 BCJR Decoding

- Determine the probability of each ending state.



- Probability of ending the trellis transition ($\Omega \rightarrow \Omega'$) at state Ω' (Determined by a backward trace).

$$B_{t'+1}(\Omega') = N_B \sum_{(\Omega'', \Omega''')} B_{t'+2}(\Omega''') \cdot \Gamma_{\Omega' \rightarrow \Omega'''} \cdot$$

N_B : Normalization factor that ensures

$$B_{t'+1}(a) + B_{t'+1}(b) + B_{t'+1}(c) + B_{t'+1}(d) = 1.$$

- By ensuring after encoding, the shift registers (encoder) are restored to the all zero state (achieved by bit tailing), we can initialize:

$$B_{k+2}(a) = 1, \text{ and } B_{k+2}(b) = B_{k+2}(c) = B_{k+2}(d) = 0.$$

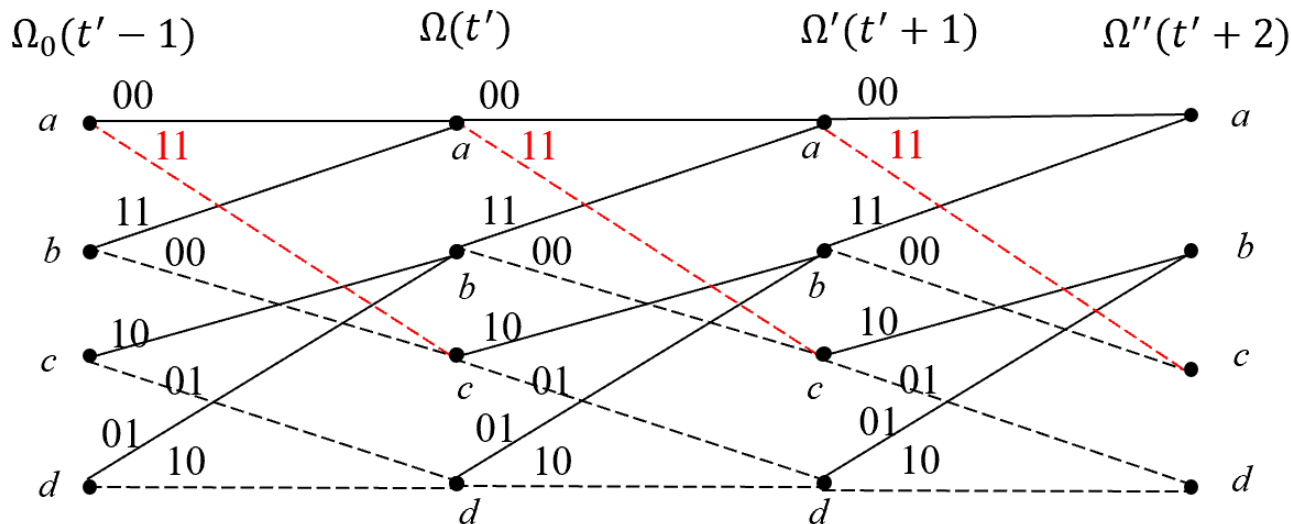
- E.g., in the highlighted trellis transition

$$B_{t'+1}(c) = B_{t'+2}(b) \cdot \Gamma_{c \rightarrow b} + B_{t'+2}(d) \cdot \Gamma_{c \rightarrow d}.$$



§ 5.5 BCJR Decoding

- Determine the *a posteriori* probability of each information bit



After the **Forward Trace** and **Backward Trace**, we obtain all the $A_{t'}(\Omega)$, $B_{t'+1}(\Omega')$ and $\Gamma_{\Omega \rightarrow \Omega'}$ of each time instant t' . We can now determine the *a posteriori* probabilities $P(u_{t'}|y_t)$ for each information bit as

$$P(u_{t'} = 0|y_t) = N_P \sum_{(\Omega, \Omega')_0} A_{t'}(\Omega) \cdot \Gamma_{\Omega \rightarrow \Omega'} \cdot B_{t'+1}(\Omega')$$

\uparrow State transition indicated by ———

$$P(u_{t'} = 1|y_t) = N_P \sum_{(\Omega, \Omega')_1} A_{t'}(\Omega) \cdot \Gamma_{\Omega \rightarrow \Omega'} \cdot B_{t'+1}(\Omega')$$

\uparrow State transition indicated by - - - -

N_P : Normalization factor that ensures $P(u_{t'} = 0|y_t) + P(u_{t'} = 1|y_t) = 1$



§ 5.5 BCJR Decoding

- E.g.,

$$P(u_{t'} = 0|y_t) = N_p \cdot (A_{t'}(a) \cdot \Gamma_{a \rightarrow a} \cdot B_{t'+1}(a) + A_{t'}(b) \cdot \Gamma_{b \rightarrow a} \cdot B_{t'+1}(a) \\ A_{t'}(c) \cdot \Gamma_{c \rightarrow b} \cdot B_{t'+1}(b) + A_{t'}(d) \cdot \Gamma_{d \rightarrow b} \cdot B_{t'+1}(b)).$$

$$N_p = P(u_{t'} = 0|y_t) + P(u_{t'} = 1|y_t)$$

- Decision based on the *a posteriori* probabilities.

$$\hat{u}_{t'} = 0, \text{ if } P(u_{t'} = 0|y_t) \geq P(u_{t'} = 1|y_t)$$

$$\hat{u}_{t'} = 1, \text{ if } P(u_{t'} = 1|y_t) > P(u_{t'} = 0|y_t)$$

Similarly, $P(c_{t'}^1 = 0|y_t)$, $P(c_{t'}^1 = 1|y_t)$, $P(c_{t'}^2 = 0|y_t)$, $P(c_{t'}^2 = 1|y_t)$ can be made.



§ 5.5 BCJR Decoding

Example 5.7. With the same transmitted codeword and received symbols of *Example 5.6*, use the BCJR algorithm to decode it.

With the received symbols, we can determine

$$\begin{cases} P_{ch}(c_1^1 = 0) = 0.83 \\ P_{ch}(c_1^1 = 1) = 0.17 \end{cases} \quad \begin{cases} P_{ch}(c_1^2 = 0) = 0.92 \\ P_{ch}(c_1^2 = 1) = 0.08 \end{cases} \quad \begin{cases} P_{ch}(c_2^1 = 0) = 0.07 \\ P_{ch}(c_2^1 = 1) = 0.93 \end{cases} \quad \begin{cases} P_{ch}(c_2^2 = 0) = 0.14 \\ P_{ch}(c_2^2 = 1) = 0.86 \end{cases}$$

$$\begin{cases} P_{ch}(c_3^1 = 0) = 0.27 \\ P_{ch}(c_3^1 = 1) = 0.73 \end{cases} \quad \begin{cases} P_{ch}(c_3^2 = 0) = 0.12 \\ P_{ch}(c_3^2 = 1) = 0.88 \end{cases} \quad \begin{cases} P_{ch}(c_4^1 = 0) = 0.90 \\ P_{ch}(c_4^1 = 1) = 0.10 \end{cases} \quad \begin{cases} P_{ch}(c_4^2 = 0) = 0.20 \\ P_{ch}(c_4^2 = 1) = 0.80 \end{cases}$$

$$\begin{cases} P_{ch}(c_5^1 = 0) = 0.92 \\ P_{ch}(c_5^1 = 1) = 0.08 \end{cases} \quad \begin{cases} P_{ch}(c_5^2 = 0) = 0.86 \\ P_{ch}(c_5^2 = 1) = 0.14 \end{cases} \quad \begin{cases} P_{ch}(c_6^1 = 0) = 0.14 \\ P_{ch}(c_6^1 = 1) = 0.86 \end{cases} \quad \begin{cases} P_{ch}(c_6^2 = 0) = 0.88 \\ P_{ch}(c_6^2 = 1) = 0.12 \end{cases}$$

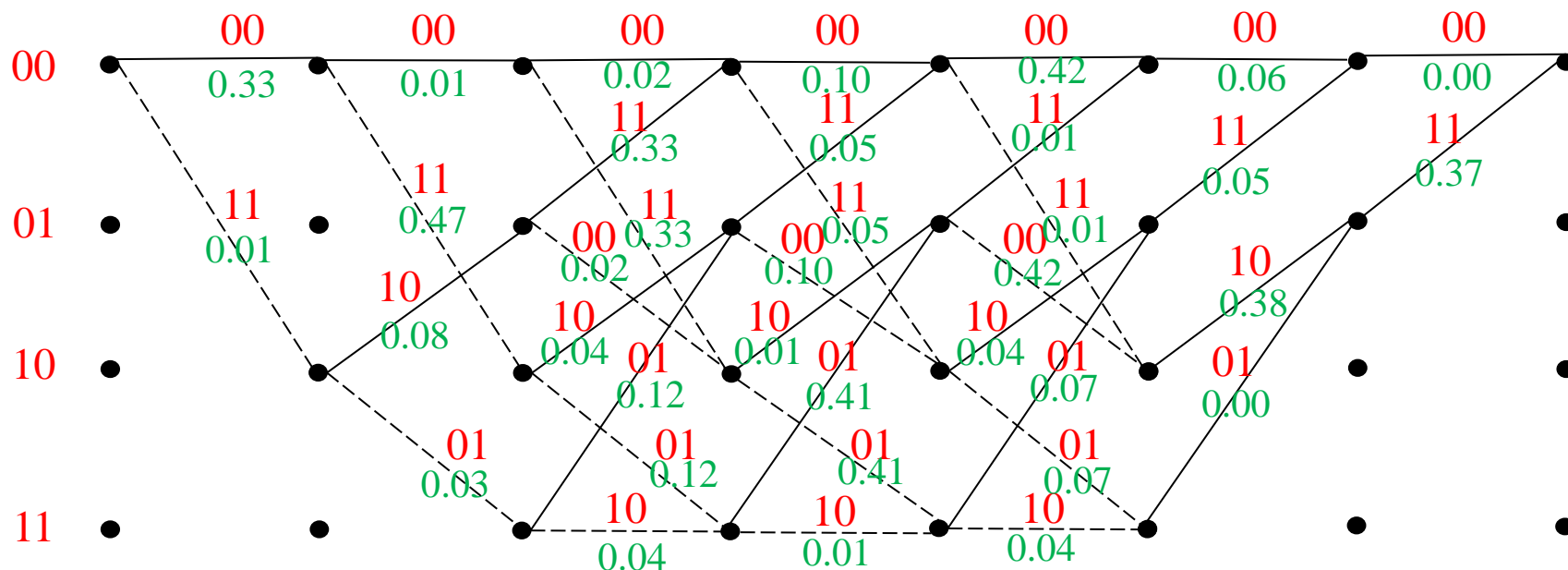
$$\begin{cases} P_{ch}(c_7^1 = 0) = 0.10 \\ P_{ch}(c_7^1 = 1) = 0.90 \end{cases} \quad \begin{cases} P_{ch}(c_7^2 = 0) = 0.17 \\ P_{ch}(c_7^2 = 1) = 0.83 \end{cases}$$



§ 5.5 BCJR Decoding

Step 1: Determine $\Gamma_{\Omega \rightarrow \Omega'}$ of all transitions.

$(0.8, 0.2)$ $(-1.3, 0.3)$ $(-0.5, 0.4)$ $(1.1, 0.4)$ $(1.2, 0.2)$ $(-0.9, -0.2)$ $(-1.1, 0)$
 $(1.2, -0.4)$ $(-0.9, -0.1)$ $(-1.0, 0.1)$ $(-0.7, -0.2)$ $(0.9, 0.3)$ $(1, 0.2)$ $(-0.8, 0.1)$

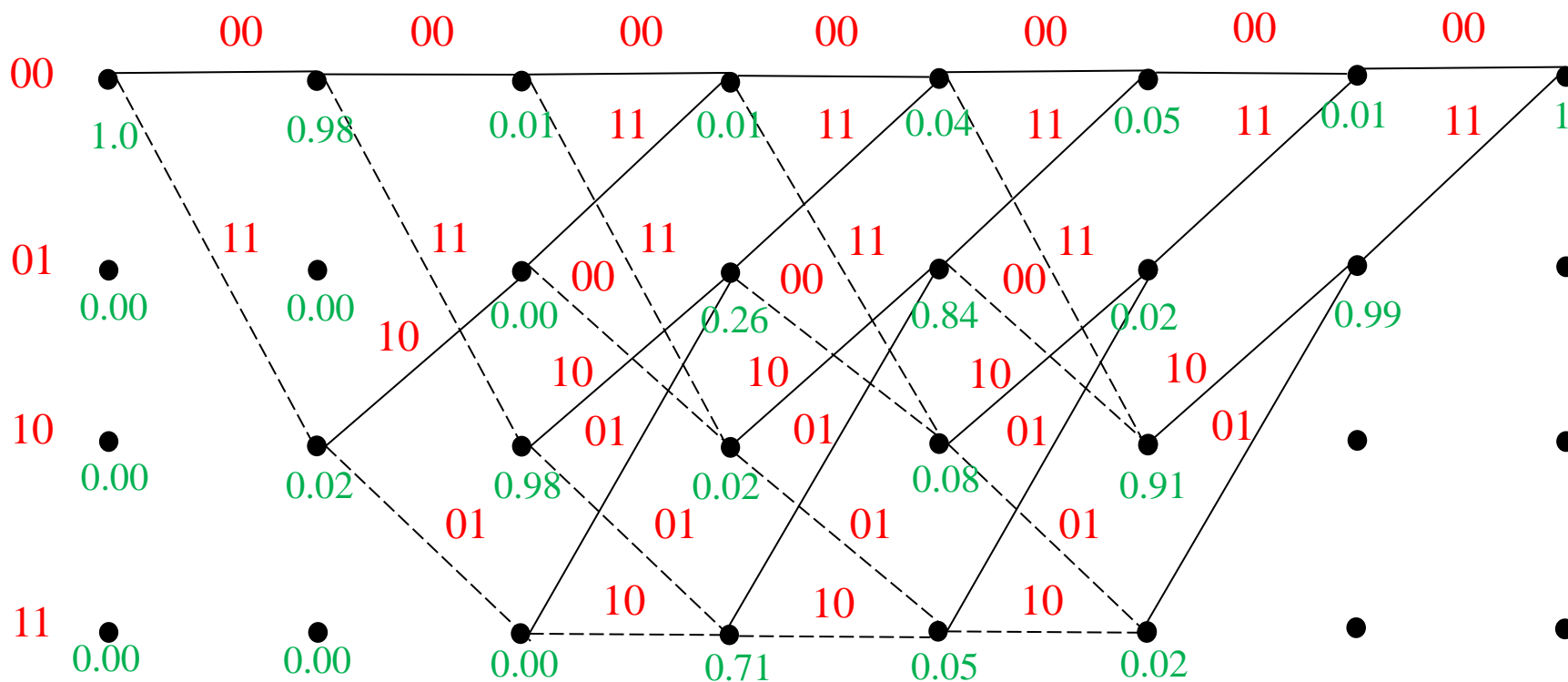




§ 5.5 BCJR Decoding

Step 2: Forward trace, determine $A_{t'}(\Omega)$ of values.

(0.8, 0.2) (-1.3, 0.3) (-0.5, 0.4) (1.1, 0.4) (1.2, 0.2) (-0.9, -0.2) (-1.1, 0)
 (1.2, -0.4) (-0.9, -0.1) (-1.0, 0.1) (-0.7, -0.2) (0.9, 0.3) (1, 0.2) (-0.8, 0.1)

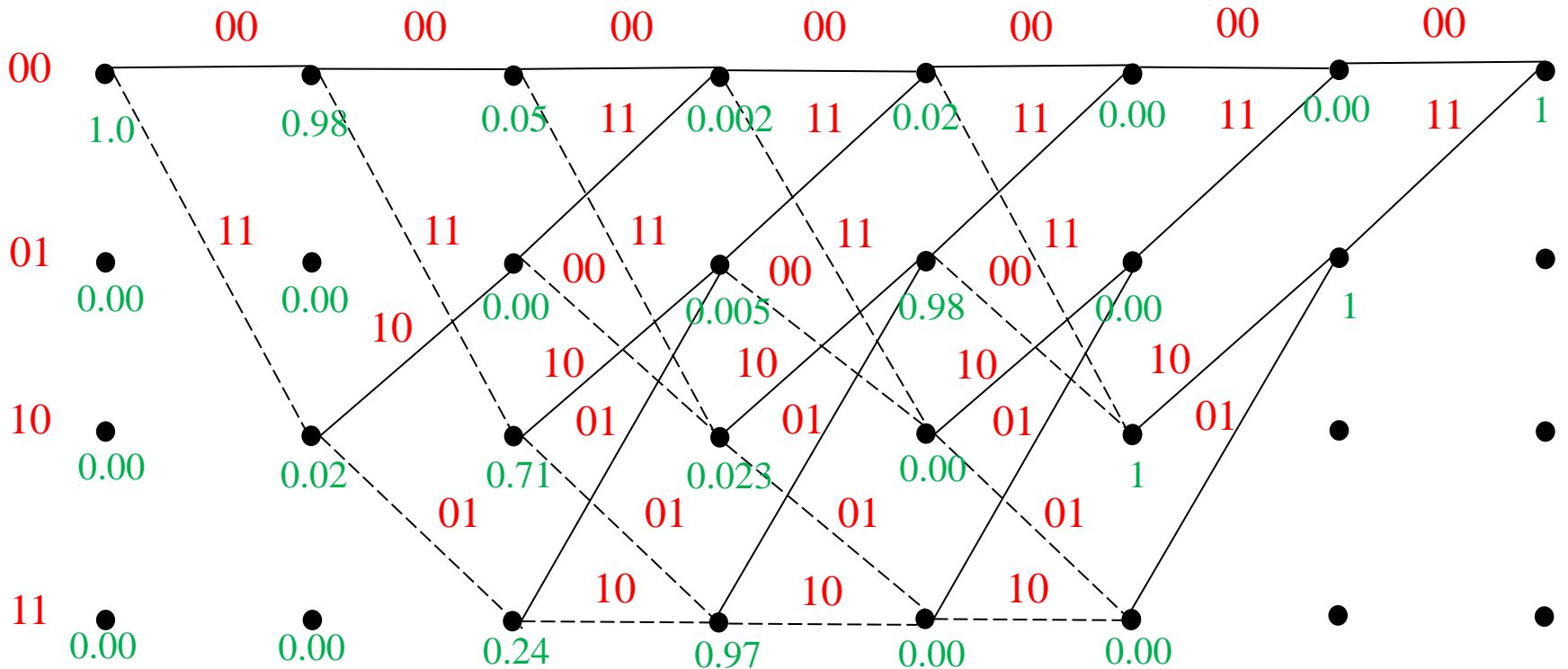




§ 5.5 BCJR Decoding

Step 3: Backward Trace, determine $B_{t'+1}(\Omega')$ of values.

(0.8, 0.2) (-1.3, 0.3) (-0.5, 0.4) (1.1, 0.4) (1.2, 0.2) (-0.9, -0.2) (-1.1, 0)
 (1.2, -0.4) (-0.9, -0.1) (-1.0, 0.1) (-0.7, -0.2) (0.9, 0.3) (1, 0.2) (-0.8, 0.1)





§ 5.5 BCJR Decoding

Step 4: Determine the *a posteriori* probabilities of each information bit.

$$\left\{ \begin{array}{l} P(u_1 = 0|y_1) = 1 \\ P(u_1 = 1|y_1) = 0 \end{array} \right. \Rightarrow \hat{u}_1 = 0 \qquad \left\{ \begin{array}{l} P(u_4 = 0|y_4) = 1 \\ P(u_4 = 1|y_4) = 0 \end{array} \right. \Rightarrow \hat{u}_4 = 0$$

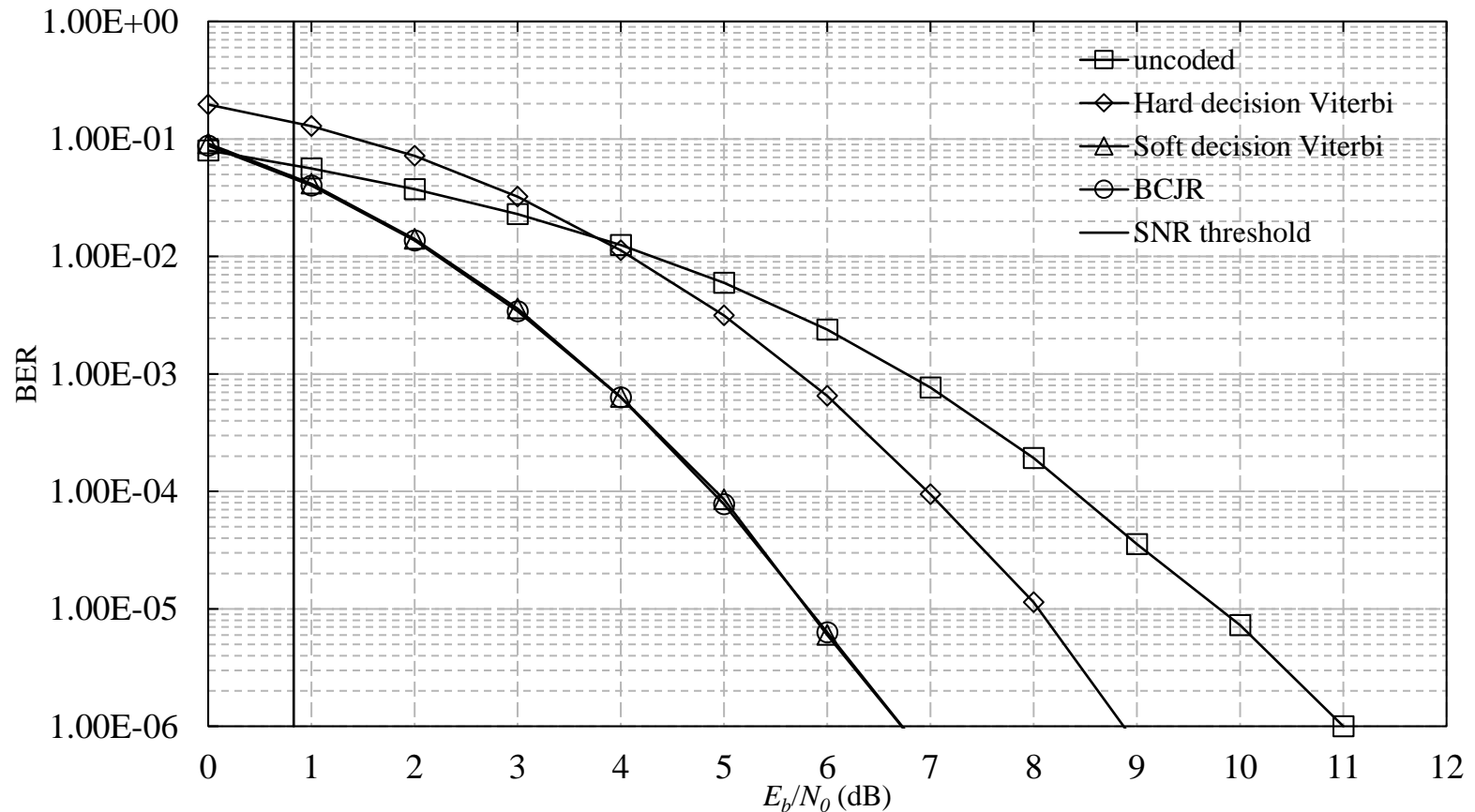
$$\left\{ \begin{array}{l} P(u_2 = 0|y_2) = 0 \\ P(u_2 = 1|y_2) = 1 \end{array} \right. \Rightarrow \hat{u}_2 = 1 \qquad \left\{ \begin{array}{l} P(u_5 = 0|y_5) = 0 \\ P(u_5 = 1|y_5) = 1 \end{array} \right. \Rightarrow \hat{u}_5 = 1$$

$$\left\{ \begin{array}{l} P(u_3 = 0|y_3) = 0 \\ P(u_3 = 1|y_3) = 1 \end{array} \right. \Rightarrow \hat{u}_3 = 1 \qquad \hat{u}_6 = \hat{u}_7 = 0$$



§ 5.5 BCJR Decoding

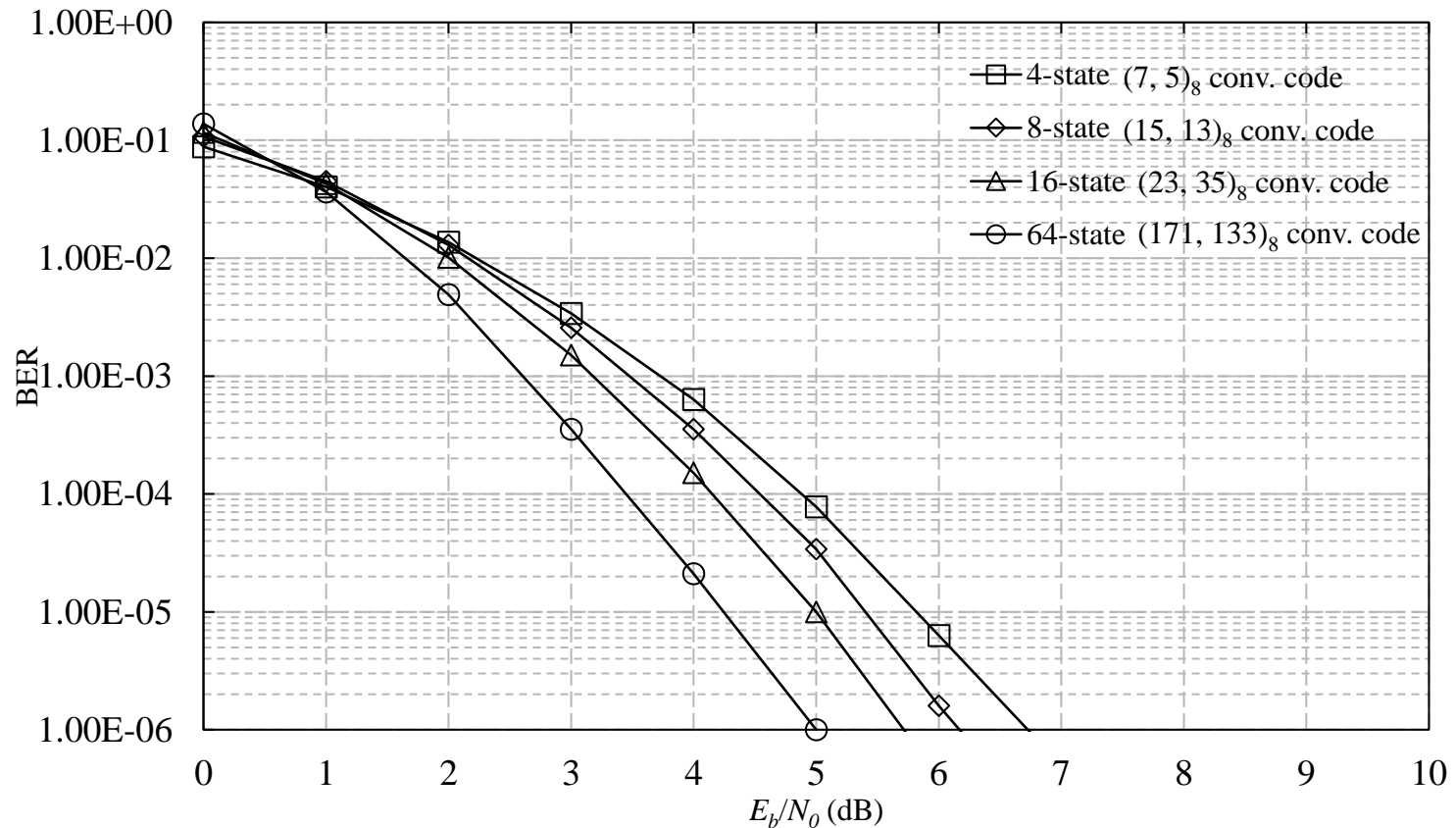
BER performance of $(7, 5)_8$ conv. code over AWGN channel using BPSK.





§ 5.5 BCJR Decoding

BER performance of different conv. code over AWGN channel using BPSK.





§ 5.6 Trellis Coded Modulation

- Convolutional code enables reliable communications. But as a channel code, its error-correction function is on the expense of spectral efficiency.

- Spectral efficiency (η) = $\frac{\text{Nr. of information bits}}{\text{transmitted symbol}}$

- E.g., an uncoded system using BPSK

$$\eta = 1 \text{ info bits/symbol}$$

- A rate 1/2 conv. coded system using BPSK

$$\eta = 0.5 \text{ info bits/symbol}$$

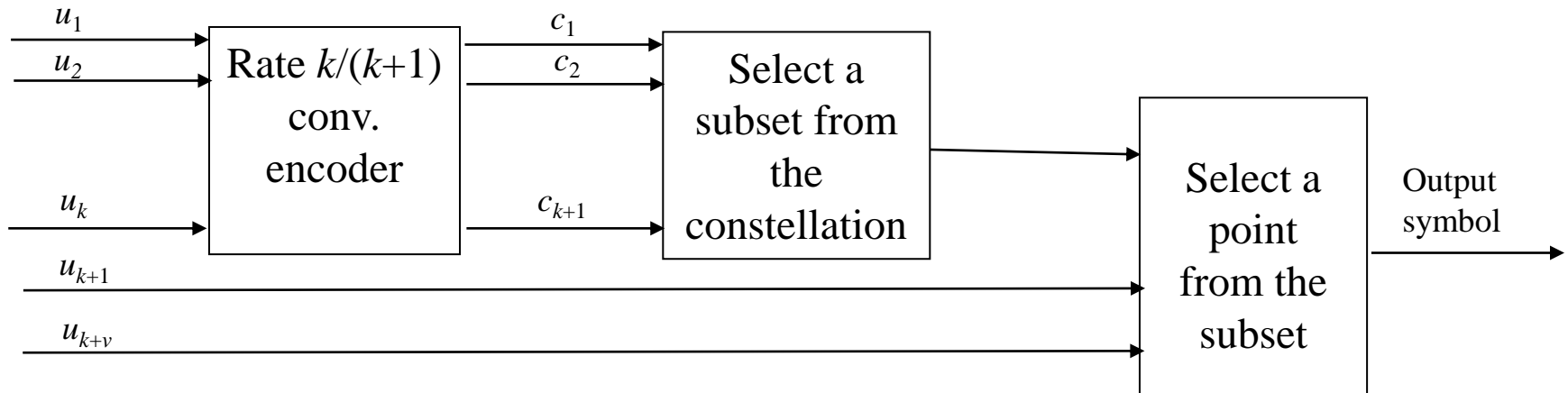
- Can we achieve reliable and yet spectrally efficient communication?

Solution: Trellis Coded Modulation (TCM) that integrates a conv. code with a high order modulation [3].



§ 5.6 Trellis Coded Modulation

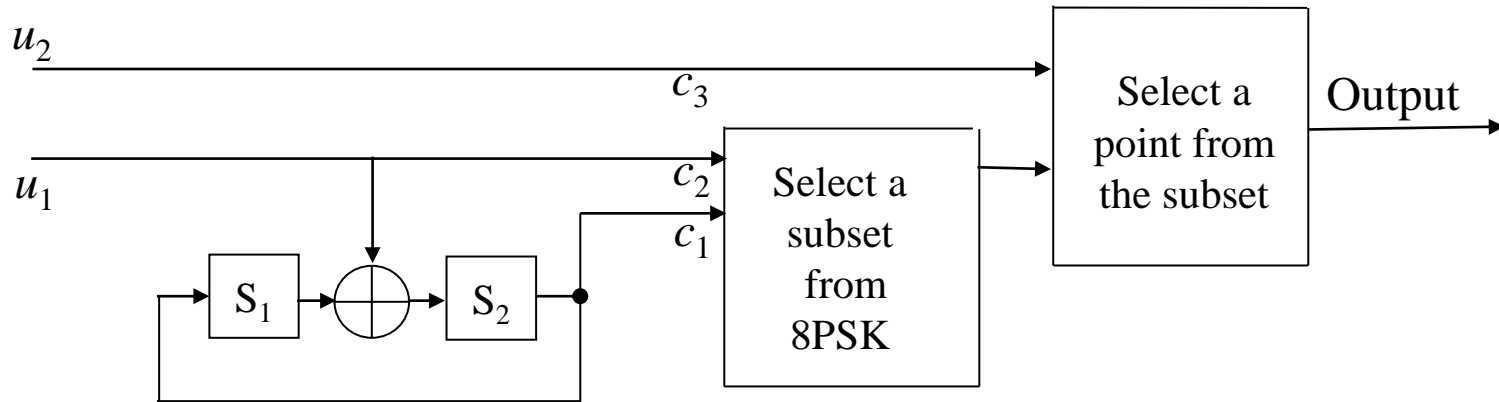
- A general structure of the TCM scheme





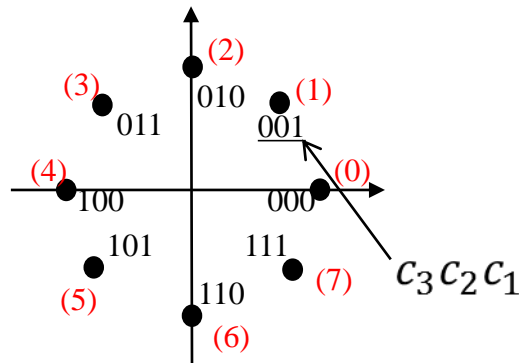
§ 5.6 Trellis Coded Modulation

- A rate 2/3 TCM code.



Rate 1/2 4-state Convolutional Code

8PSK Constellation





§ 5.6 Trellis Coded Modulation

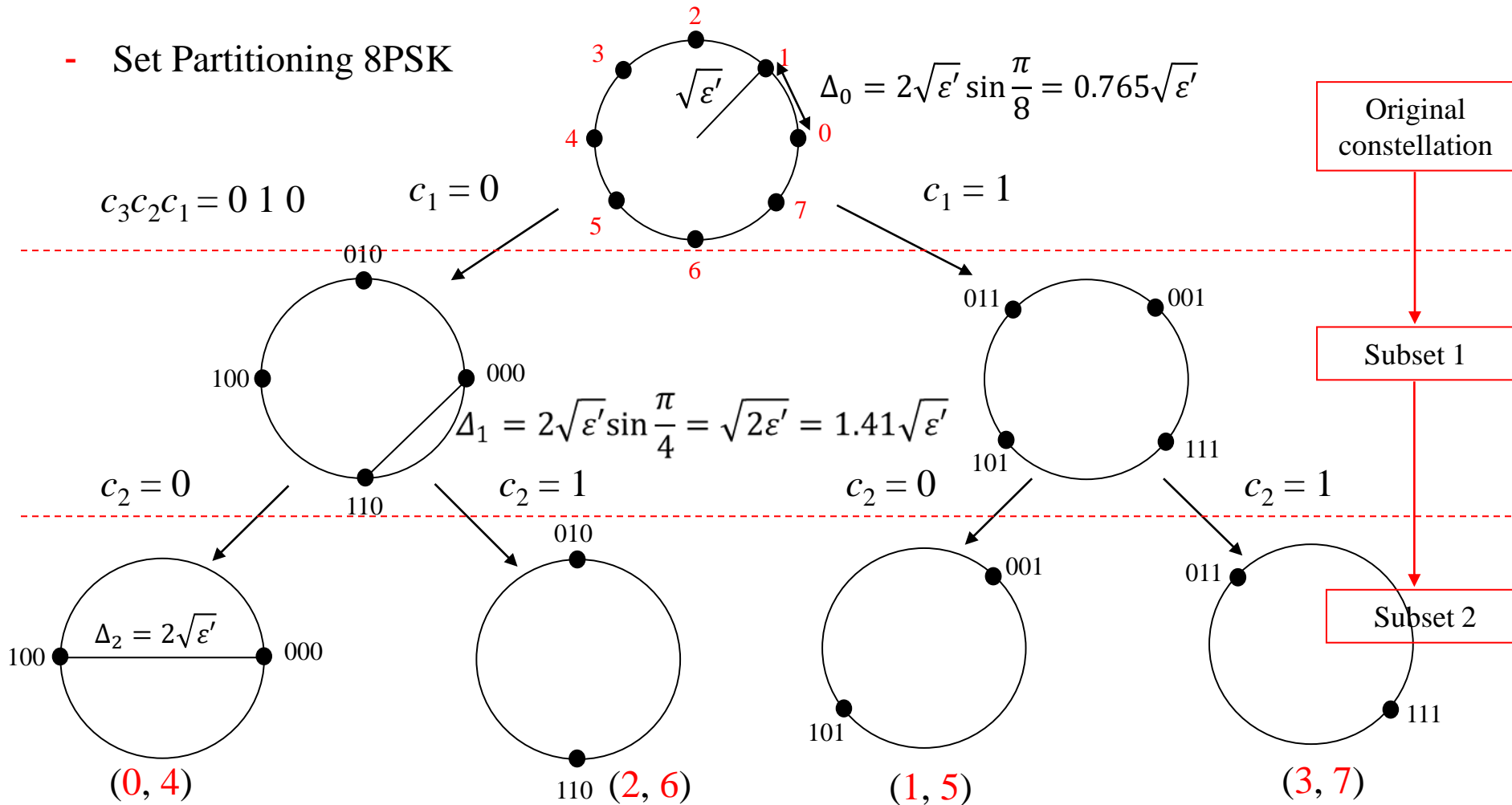
- State table of the rate 2/3 TCM code

Input		Current State		Next State		Output			Symbol
u_1	u_2	S_1	S_2	S_1'	S_2'	c_1	c_2	c_3	8PSK sym
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	1	0	2
0	1	0	0	0	0	0	0	1	4
1	1	0	0	0	1	0	1	1	6
0	0	0	1	1	0	1	0	0	1
1	0	0	1	1	1	1	1	0	3
0	1	0	1	1	0	1	0	1	5
1	1	0	1	1	1	1	1	1	7
0	0	1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	1	0	2
0	1	1	0	0	1	0	0	1	4
1	1	1	0	0	0	0	1	1	6
0	0	1	1	1	1	1	0	0	1
1	0	1	1	1	0	1	1	0	3
0	1	1	1	1	1	1	0	1	5
1	1	1	1	1	0	1	1	1	7



§ 5.6 Trellis Coded Modulation

- Set Partitioning 8PSK

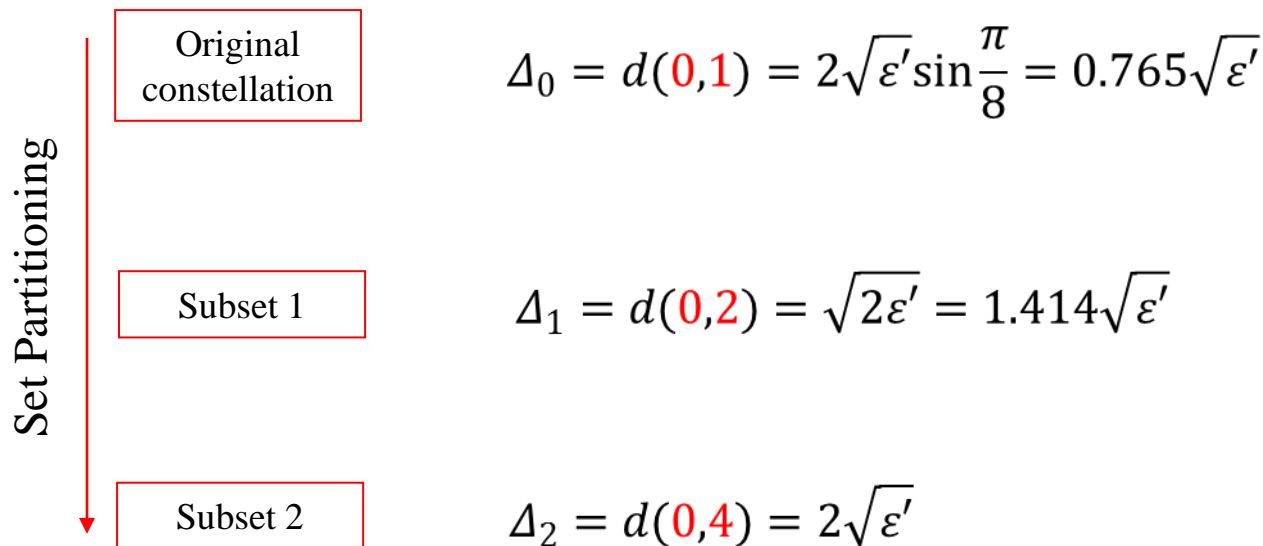




§ 5.6 Trellis Coded Modulation

- Set Partitioning 8PSK

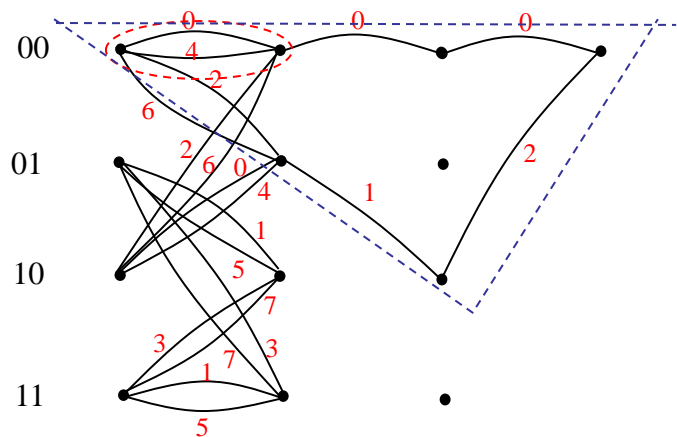
By doing set partitioning, the minimum distance between point within a subset is increasing as: $\Delta_0 < \Delta_1 < \Delta_2$.





§ 5.6 Trellis Coded Modulation

- Viterbi trellis of the rate 2/3 TCM code



For diverse/remerge transition:

$$d_{\text{free}}^2 = [d^2(0,2) + d^2(0,1) + d^2(0,2)]$$

$$= 2\varepsilon' + (2 - \sqrt{2})\varepsilon' + 2\varepsilon' = 4.568\varepsilon'$$

For parallel transition:

$$d_{\text{free}}^2 = d^2(0,4) = 4\varepsilon'$$

Choose the smaller one as the free distance of the code:

$$d_{\text{free}}^2 = 4\varepsilon'$$

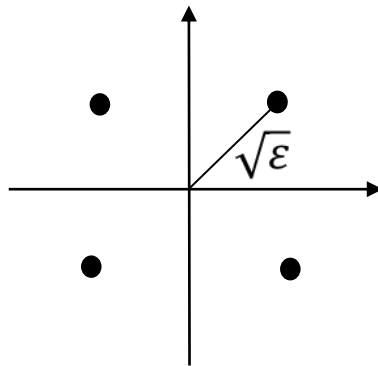
- Bit $c_3 = 0$ and $c_3 = 1$ result in two parallel transition branches. By doing set partitioning, we are trying to maximize the Euclidean distance between the two parallel branches. So that the free distance of the TCM code can be maximized.



§ 5.6 Trellis Coded Modulation

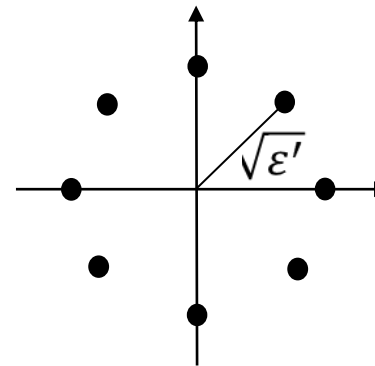
- Asymptotic coding gain over an uncoded system.
- Spectral efficiency (η) = 2 info bits/sym.

uncoded QPSK



$$d_{min}^2 = 2\epsilon$$
$$\text{Asymptotic coding gain } \gamma = \frac{(d_{free}^2/\epsilon')}{(d_{min}^2/\epsilon)} = 2.$$

rate 2/3 coded 8PSK



$$d_{free}^2 = 4\epsilon'$$

Asymptotic coding gain in dB = $10 \log_{10} \gamma = 3$ dB.

- With the same transmission spectral efficiency of 2 info bits/sym, the TCM coded system achieves 3 dB coding gain over the uncoded system asymptotically.



§ 5.7* Sequential Decoding

- Introduction

- Sequential decoding [1] is a suboptimal decoding approach elaborated over the binary tree, circumventing the high complexity in Viterbi decoding of conv. codes with a large memory m .
- By setting a dynamic threshold adjusted during the decoding, it compares different paths and selects the most likely one. Meanwhile, it minimizes the unnecessary node search.
- For practical use, code rate must be strictly below channel capacity.
- Fano decoding [2] is a depth-first-search (DFS) algorithm. By moving forward or backward over the tree, it aims to find the most likely path. In order to rationalize the decoding effort, a complexity budget can be imposed. That says the decoding will be terminated once the budget is reached.

[1] J. M. Wozencraft, "Sequential decoding for reliable communication," *Res. Lab. Elect.*, MIT, Cambridge, MA, USA, Tech. Rep. 325, Aug. 1957.

[2] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Inf. Theory*, vol. IT-9, no. 2, pp. 64–74, Apr. 1963.



§ 5.7* Sequential Decoding

- Let us denote codeword

$$\bar{c} = (c_1, c_2, \dots, c_N) = ([\bar{c}]_t), \text{ where } t = 1, 2, \dots, N,$$

and a received symbol vector

$$\bar{y} = (y_1, y_2, \dots, y_N) = ([\bar{y}]_t), \text{ where } t = 1, 2, \dots, N.$$

- MAP decoding yield codeword $\bar{c}_{i'}$, s.t.

$$P(\bar{c}_{i'}|\bar{y}) \geq P(\bar{c}_i|\bar{y}), \forall i \neq i'$$

- Sequential decoding aims to find a likely codeword $\bar{c}_{i'}$, s.t. its *a posteriori* probability $P(\bar{c}_{i'}|\bar{y})$ is greater than or equal to a threshold. Relaxing the MAP criterion, let $\sum_{\substack{i=1 \\ i \neq i'}}^{2^K} P(\bar{c}_i|\bar{y})$ be the threshold. The decoding finds $\bar{c}_{i'}$, s.t.

$$P(\bar{c}_{i'}|\bar{y}) \geq \sum_{i=1, i \neq i'}^{2^K} P(\bar{c}_i|\bar{y}) \quad (1)$$



§ 5.7* Sequential Decoding

- Assumptions:
Random coding,

$$P(\bar{c}_i) = \prod_{t=1}^N P([\bar{c}_i]_t) = 2^{-K}$$

- Discrete memoryless channel,

$$P(\bar{y}|\bar{c}_i) = \prod_{t=1}^N P([\bar{y}]_t|[\bar{c}_i]_t)$$

- By Bayes' rule, we have

$$P(\bar{y}|\bar{c}_i) = \frac{P(\bar{c}_i|\bar{y}) \cdot P(\bar{y})}{P(\bar{c}_i)}, \forall i$$

- The decoding yields a codeword $\bar{c}_{i'}$ that satisfies

$$P(\bar{y}|\bar{c}_{i'}) \geq \sum_{i=1, i \neq i'}^{2^K} P(\bar{y}|\bar{c}_i) \quad (2)$$



§ 5.7* Sequential Decoding

- Let $\tilde{P}(\bar{y}|\bar{c}_i)$ denote the average of all transition probabilities $P(\bar{y}|\bar{c}_i)$ in the right hand side of (2).
- Hence

$$P(\bar{y}|\bar{c}_{i'}) \geq (2^K - 1) \cdot \tilde{P}(\bar{y}|\bar{c}_i), \quad (3)$$

where

$$\begin{aligned} \tilde{P}(\bar{y}|\bar{c}_i) &= \frac{\sum_{i=1}^{2^K-1} P(\bar{y}|\bar{c}_i)}{2^K - 1} \\ &\cong \sum_{i=1}^{2^K-1} P(\bar{y}|\bar{c}_i) \cdot P(\bar{c}_i) \\ &= \sum_{i=1}^{2^K-1} P(\bar{c}_i, \bar{y}) \\ &\cong P(\bar{y}) \end{aligned}$$



§ 5.7* Sequential Decoding

- Therefore, ineq. (3) becomes

$$P(\bar{y}|\bar{c}_{i'}) \geq (2^K - 1) \cdot P(\bar{y})$$

⇓

$$\frac{P(\bar{y}|\bar{c}_{i'})}{P(\bar{y})} \geq 2^K - 1$$

⇓

$$\prod_{t=1}^N \frac{P([\bar{y}]_t | [\bar{c}_{i'}]_t)}{P([\bar{y}]_t)} \geq 2^K - 1$$

⇓

Approximate $2^K - 1$ to 2^K

$$\log_2 \prod_{t=1}^N \frac{P([\bar{y}]_t | [\bar{c}_{i'}]_t)}{P([\bar{y}]_t)} \geq K$$

⇓

$$\sum_{t=1}^N \left(\log_2 \frac{P([\bar{y}]_t | [\bar{c}_{i'}]_t)}{P([\bar{y}]_t)} - R \right) \geq 0$$

↑ Mutual information $I([\bar{y}]_t; [\bar{c}_{i'}]_t)$



§ 5.7* Sequential Decoding

- Denote $[\bar{c}]_t = \bar{c}_t$, $[\bar{y}]_t = \bar{y}_t$, Fano branch metric is

$$\lambda_t = \log_2 \frac{P(\bar{y}_t | \bar{c}_t)}{P(\bar{y}_t)} - n_0 \cdot R$$

where at time instant t , the input information is k_0 bits and the output codeword is n_0 bits. Code rate $R = \frac{k_0}{n_0}$.

- Metric of a path with length l is

$$\Lambda = \sum_{t=1}^l \lambda_t = \sum_{t=1}^l \log_2 \frac{P(\bar{y}_t | \bar{c}_t)}{P(\bar{y}_t)} - n_0 l R$$

- The path metric Λ increases if \bar{y}_t can yield a right decision, and decreases vice versa.
- When comparing paths of different lengths, the path with the maximum path metric is considered the most likely one.



§ 5.7* Sequential Decoding

- **Fano decoding rules:**
 - **Initialization** (start from the root): Let threshold $T = 0$, step size Δ (for adjusting T). E.g., let $\Delta = 4$.
 - **Computations** (over the code's binary tree): Calculate two forward nodes' path metrics, and denote them as Λ_{\max} and Λ_{\min} , where $\Lambda_{\max} \geq \Lambda_{\min}$. Choose one of the below operations:
 - If $\Lambda_{\max} \geq T$, move forward to its child node and update path metric as Λ_{\max} ;
 - If $\Lambda_{\max} < T$, move backward to the parent node. This may happen a few times until reaching an unexpanded ancestral node with $\Lambda_{\min} > T$. The decoder then starts to compute from this node with a path metric of Λ_{\min} .
 - **Termination** (when reaches a leaf): When decoder reach the leaf node, the decoding is completed and the path outputs yield the candidate codeword.
 - When reaching a node with path metric Λ , threshold T is **adjusted** as below:
 - If current node is reached from its parent node for the first time and $\Lambda > T + \Delta$, increase T by $T + \Delta \rightarrow T$, s.t. $\Lambda - \Delta < T \leq \Lambda$.
 - If current node is reached from its child node and $\Lambda < T$, decrease T by $T - \Delta \rightarrow T$, s.t. $\Lambda - \Delta < T \leq \Lambda$.
 - If root node is reached from its Λ_{\min} child node, decrease T by $T - \Delta \rightarrow T$.



§ 5.7* Sequential Decoding

Example 5.7 Given the $(7, 5)_8$ conv. code, the message is $[m_1 m_2 m_3 m_4 m_5 m_6 m_7] = [1 0 1 1 1 0 0]$, the codeword is $[c_1^1 c_1^2 c_2^1 c_2^2 c_3^1 c_3^2 c_4^1 c_4^2 c_5^1 c_5^2 c_6^1 c_6^2 c_7^1 c_7^2] = [11 10 00 01 10 01 11]$, the received word is $[y_1^1 y_1^2 y_2^1 y_2^2 y_3^1 y_3^2 y_4^1 y_4^2 y_5^1 y_5^2 y_6^1 y_6^2 y_7^1 y_7^2] = [10 10 01 01 10 01 11]$. Thus, $n_0 = 2$, $k_0 = 1$, $R = k_0/n_0 = 0.5$. Under BSC channel, $p = 0.02$, and with uniform input such that $P(y = 0) = P(y = 1) = 0.5$.

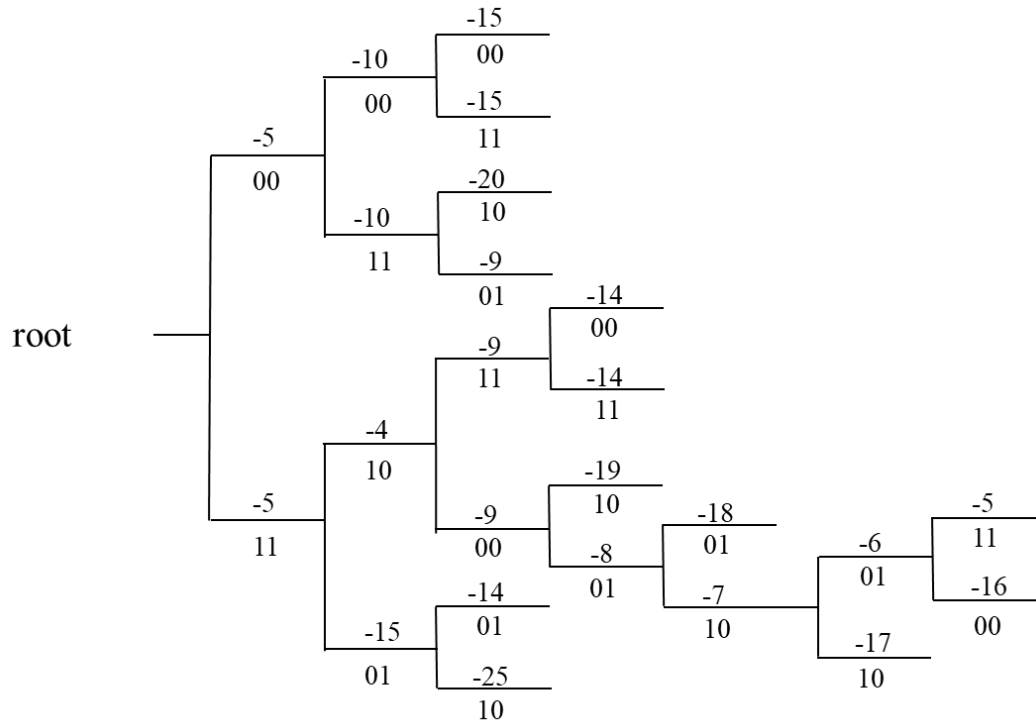
At time t , the branch metric λ_t between estimated message bit \hat{m}_t or equivalently $\hat{c}_t^1 \hat{c}_t^2$ and received symbols $y_t^1 y_t^2$ only occurs three situations as

$$\lambda_t = \log_2 \frac{P(\bar{y}_t | \bar{c}_t)}{P(\bar{y}_t)} - n_0 \cdot R = \begin{cases} 2 \cdot \log_2 \frac{1-p}{0.5} - 2 \cdot R \approx 0.94 \approx 1 & , \text{no error} \\ \log_2 \frac{1-p}{0.5} + \log_2 \frac{p}{0.5} - 2 \cdot R \approx -4.67 \approx -5 & , 1 \text{ error} \\ 2 \cdot \log_2 \frac{p}{0.5} - 2 \cdot R \approx -10.29 \approx -10 & , 2 \text{ errors} \end{cases}$$



§ 5.7* Sequential Decoding

The following graph shows the outcome if all paths have been explored



Initialization
 $T = 0, \Delta = 4$

received word	10	10	01	01	10	01	11
time t	1	2	3	4	5	6	7

↑ Input bit as 0
 ↓ Input bit as 1

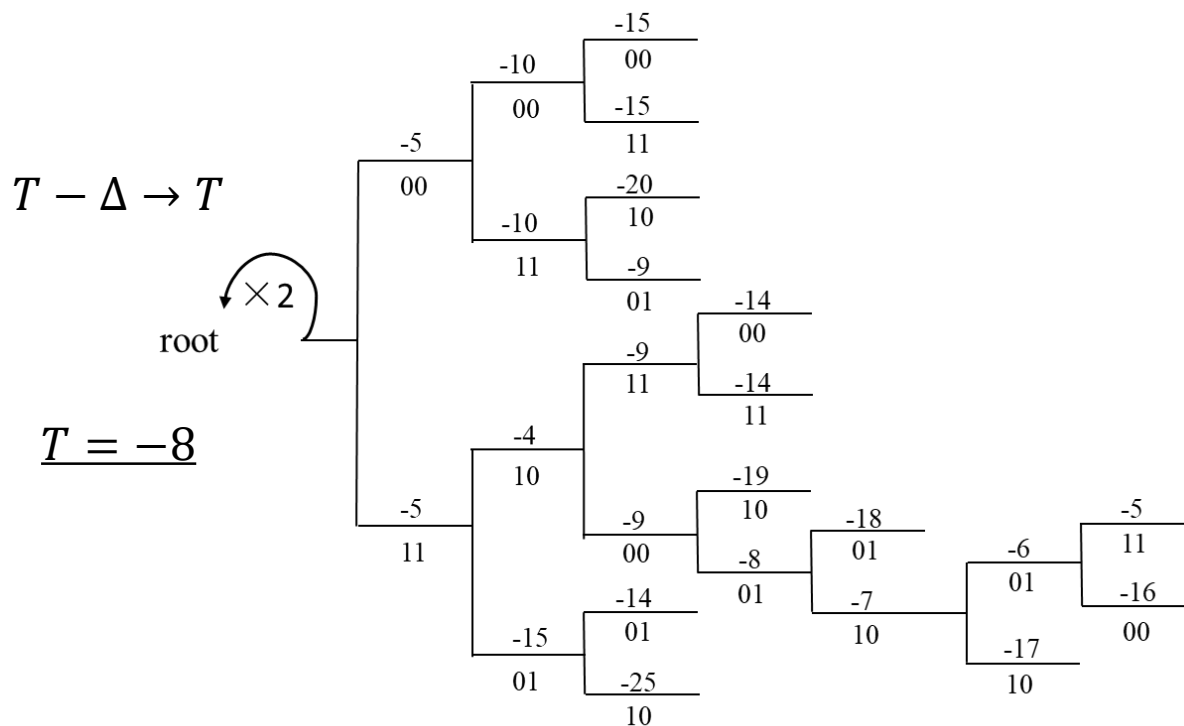
Path metric

 $\hat{c}_t^1 \hat{c}_t^2$



§ 5.7* Sequential Decoding

Calculate the path metric from the root node to the children nodes, we have $\Lambda_{\max} = -5$ and $\Lambda_{\max} < T$. Reduce threshold twice until decoder could depart from the root.

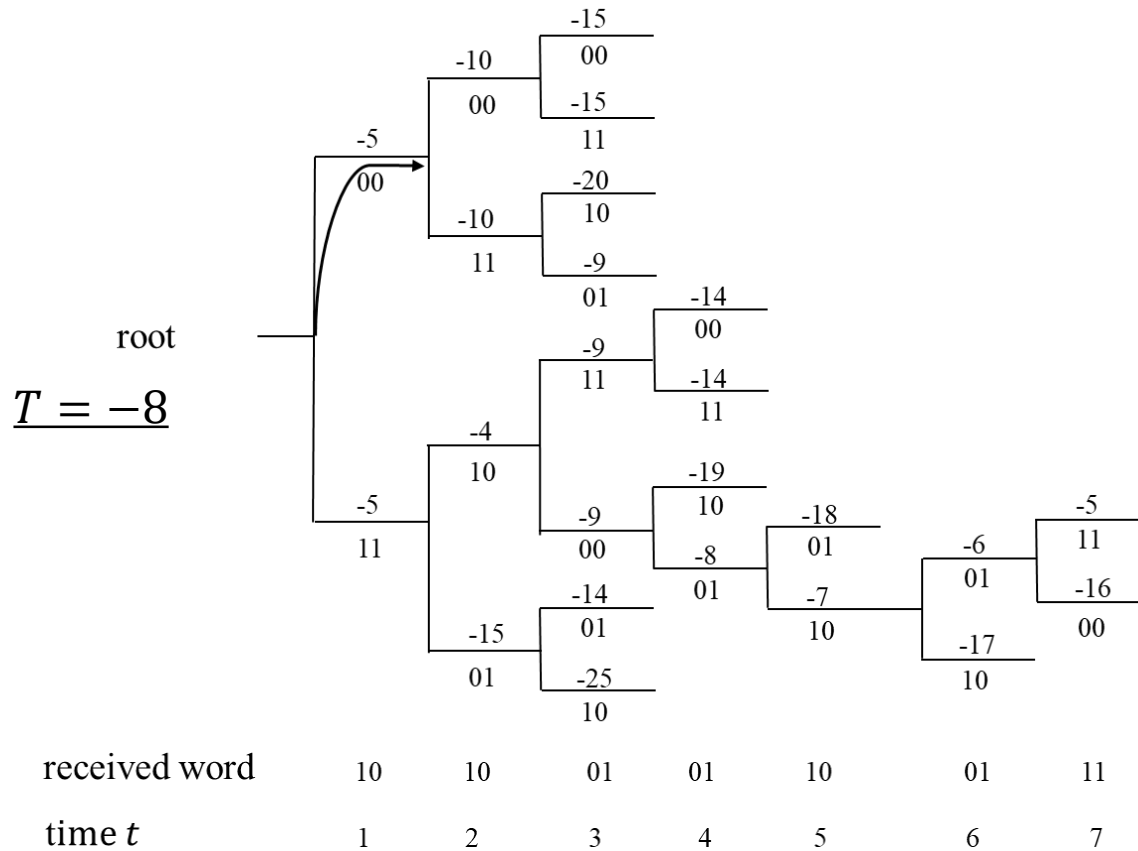


received word	10	10	01	01	10	01	11
time t	1	2	3	4	5	6	7



§ 5.7* Sequential Decoding

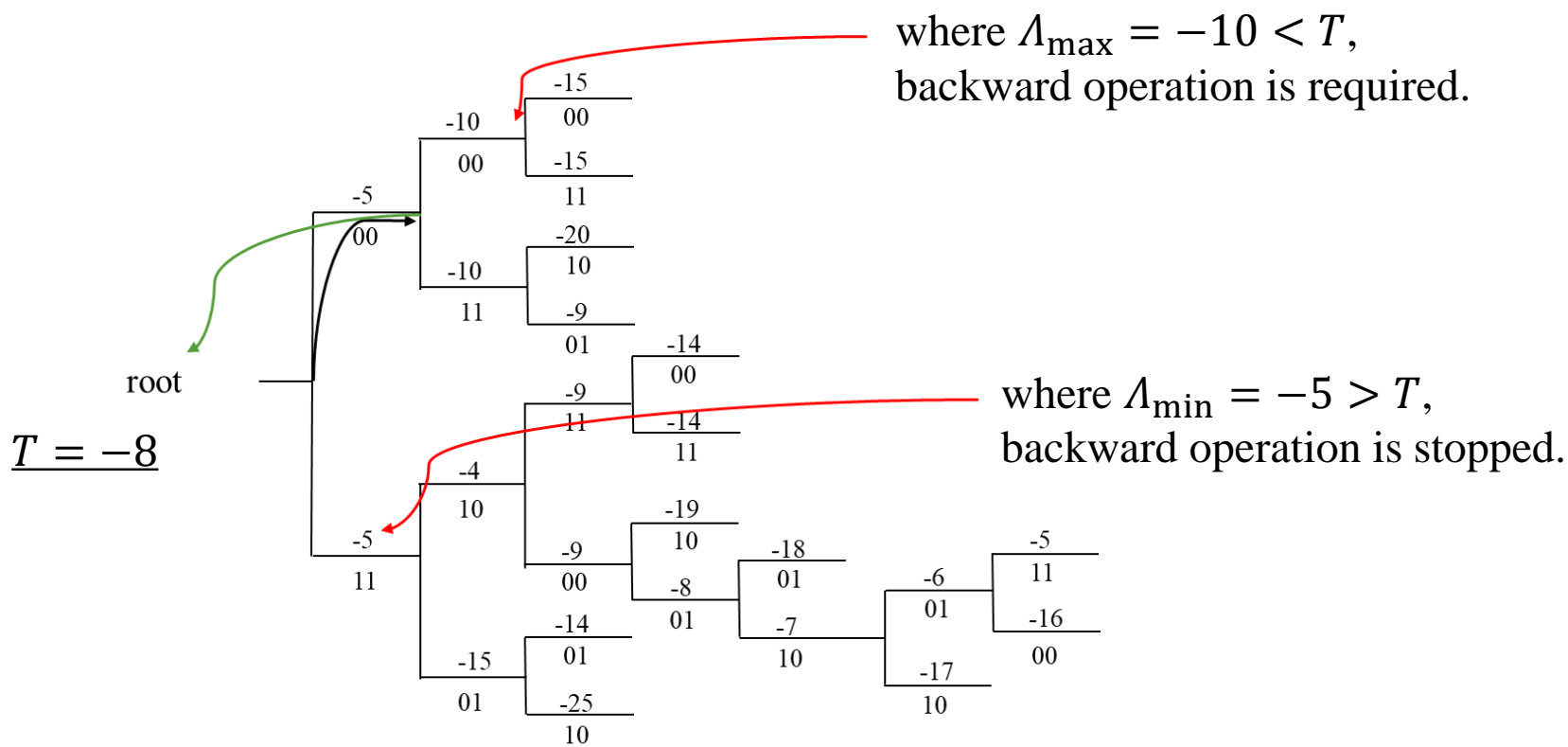
Move forward until $\Lambda_{\max} < T$.





§ 5.7* Sequential Decoding

Move backward until an unexpanded node with path metric $\Lambda_{\min} > T$.

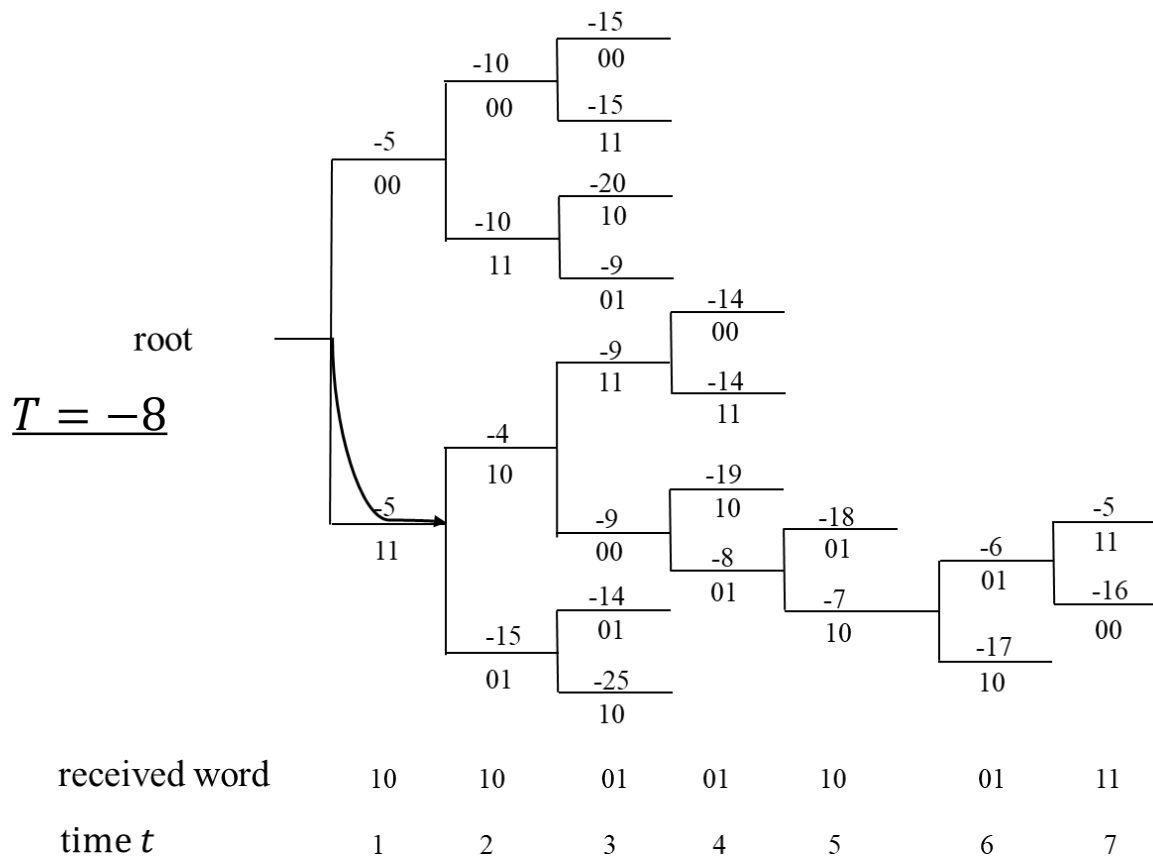


received word	10	10	01	01	10	01	11
time t	1	2	3	4	5	6	7



§ 5.7* Sequential Decoding

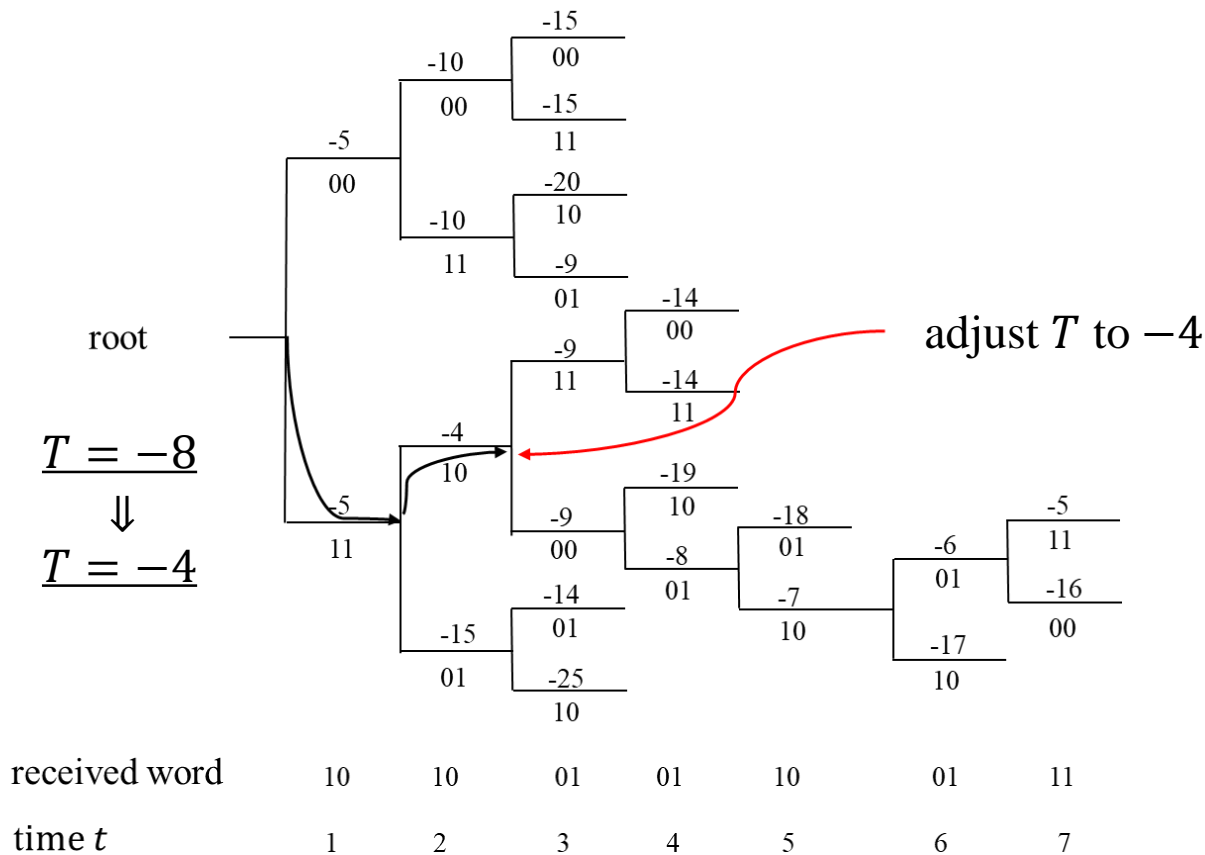
Move forward from node with path metric $\Lambda_{\min} = -5 > T$.





§ 5.7* Sequential Decoding

Move forward. If reach a node for the first time and its path metric $\Lambda = -4 \geq T + \Delta = -4$, let $T + \Delta \rightarrow T$, s.t. $\Lambda - \Delta < T \leq \Lambda$.

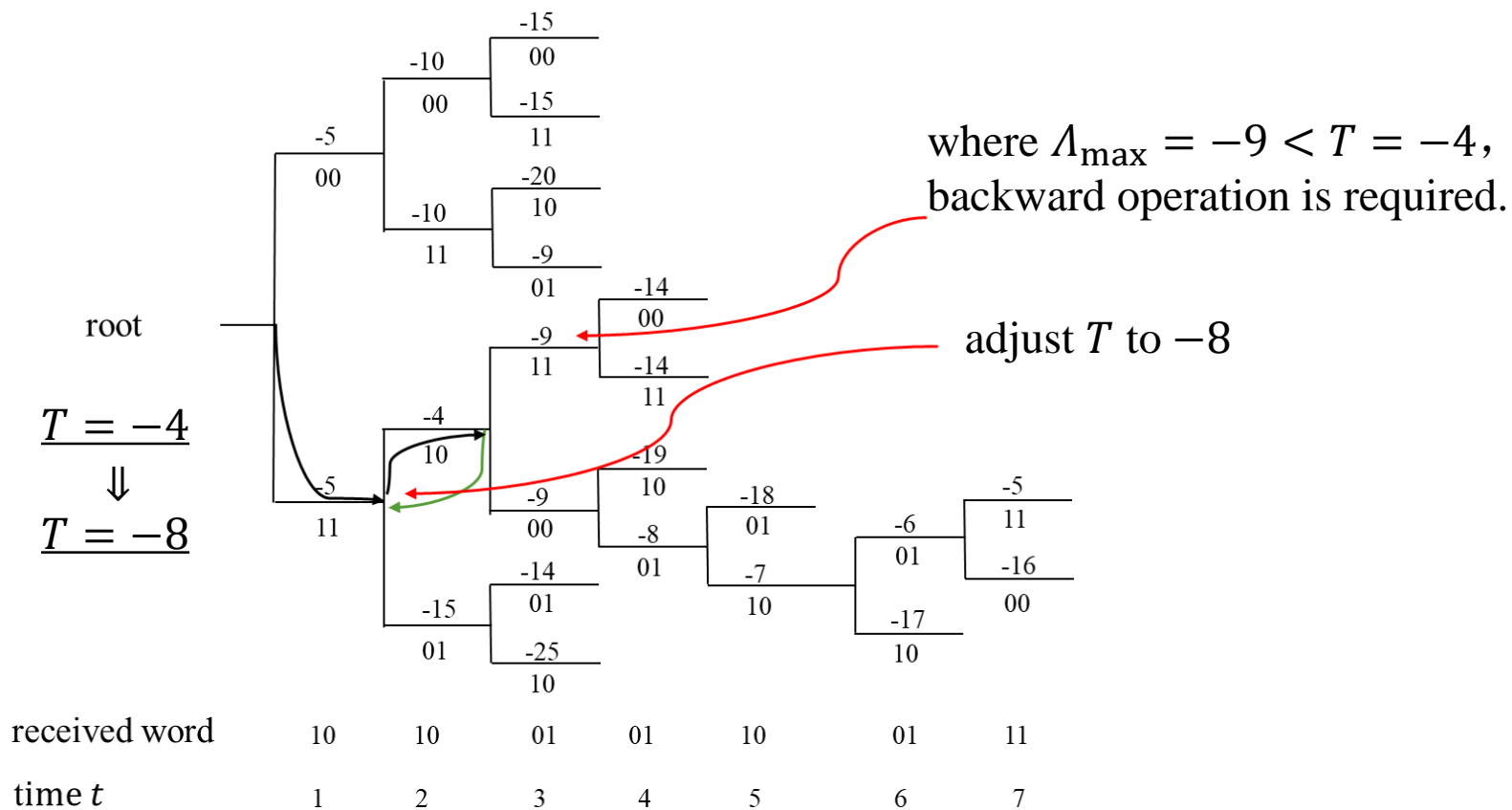




§ 5.7* Sequential Decoding

Move backward because $\Lambda_{\max} < T$.

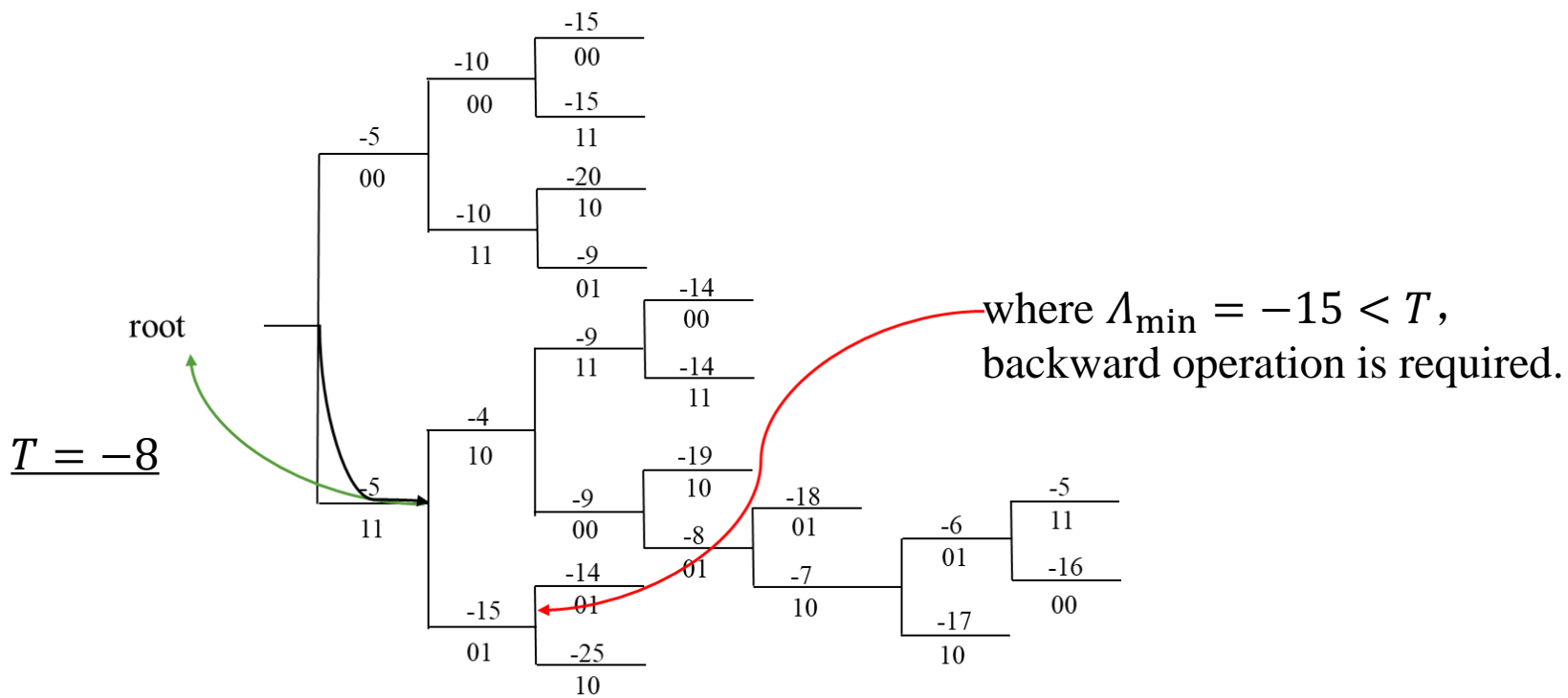
Then, it is found that $\Lambda = -5 < T = -4$. Thus, Let $T - \Delta \rightarrow T$, s.t. $\Lambda - \Delta < T \leq \Lambda$.





§ 5.7* Sequential Decoding

Keep moving backward until an unexpanded node with $\Lambda_{\min} > T$.

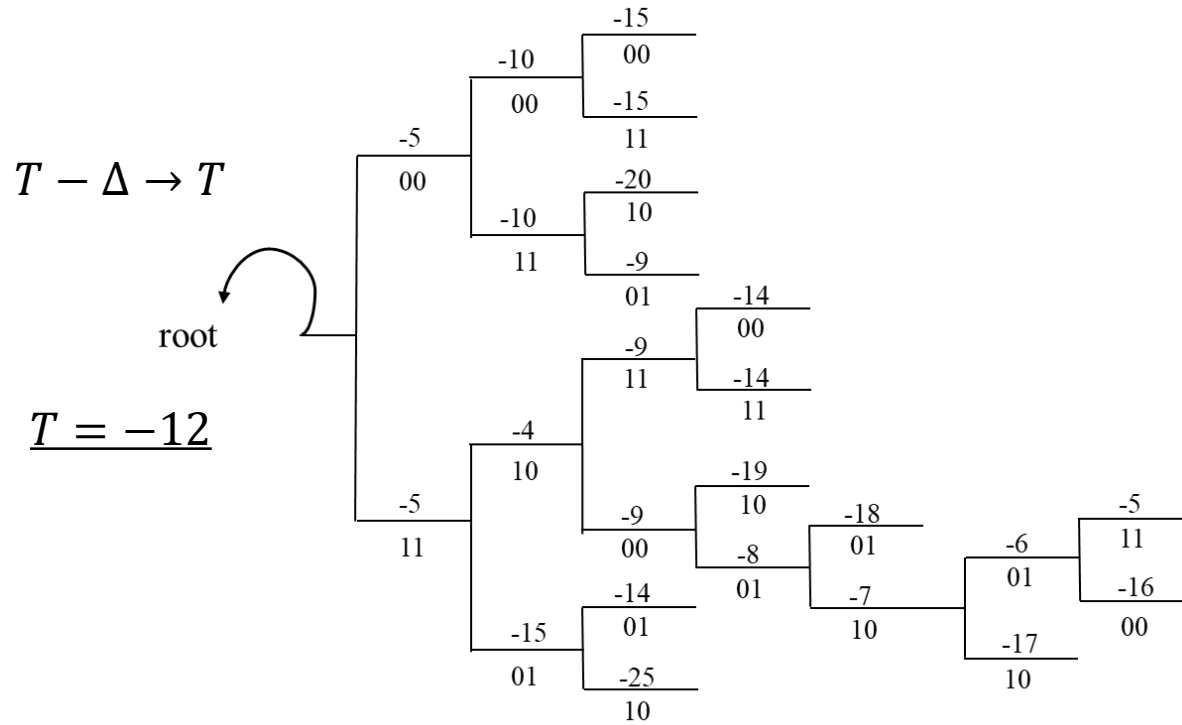


received word	10	10	01	01	10	01	11
time t	1	2	3	4	5	6	7



§ 5.7* Sequential Decoding

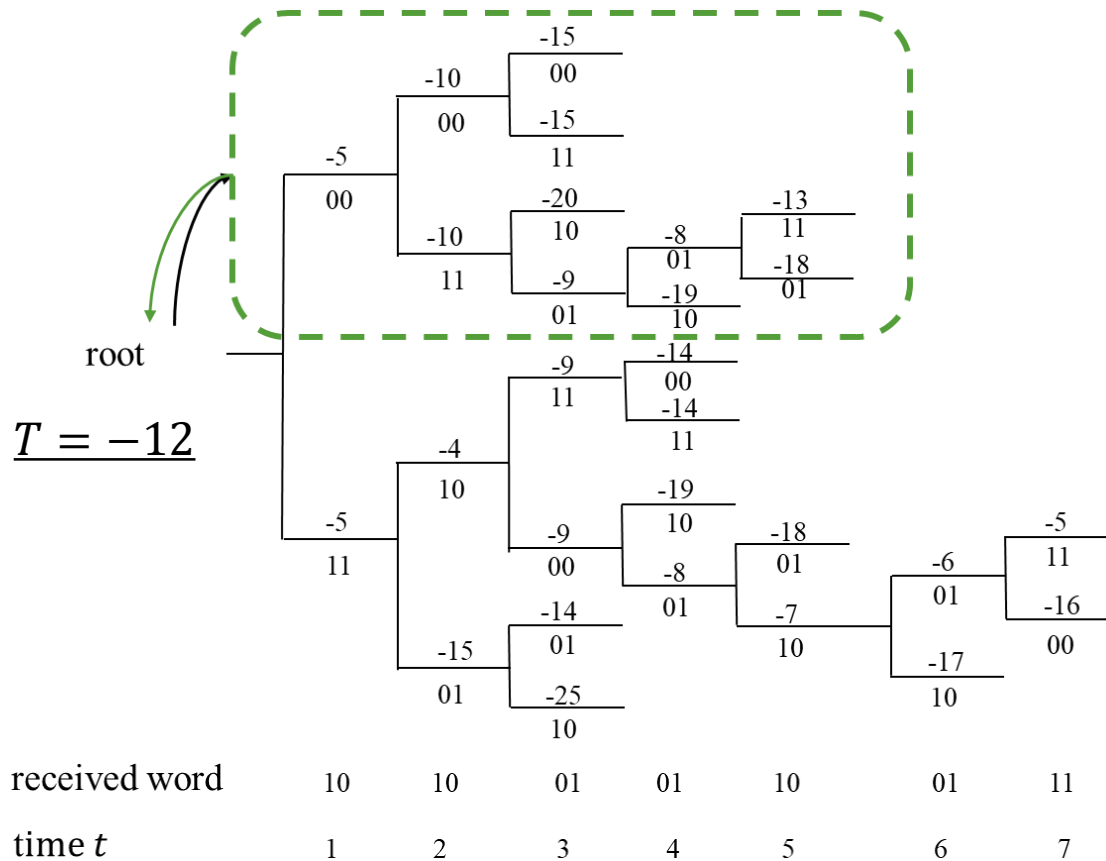
Reduce T by $T - \Delta \rightarrow T$ because root node is reached from its Λ_{\min} child node.





§ 5.7* Sequential Decoding

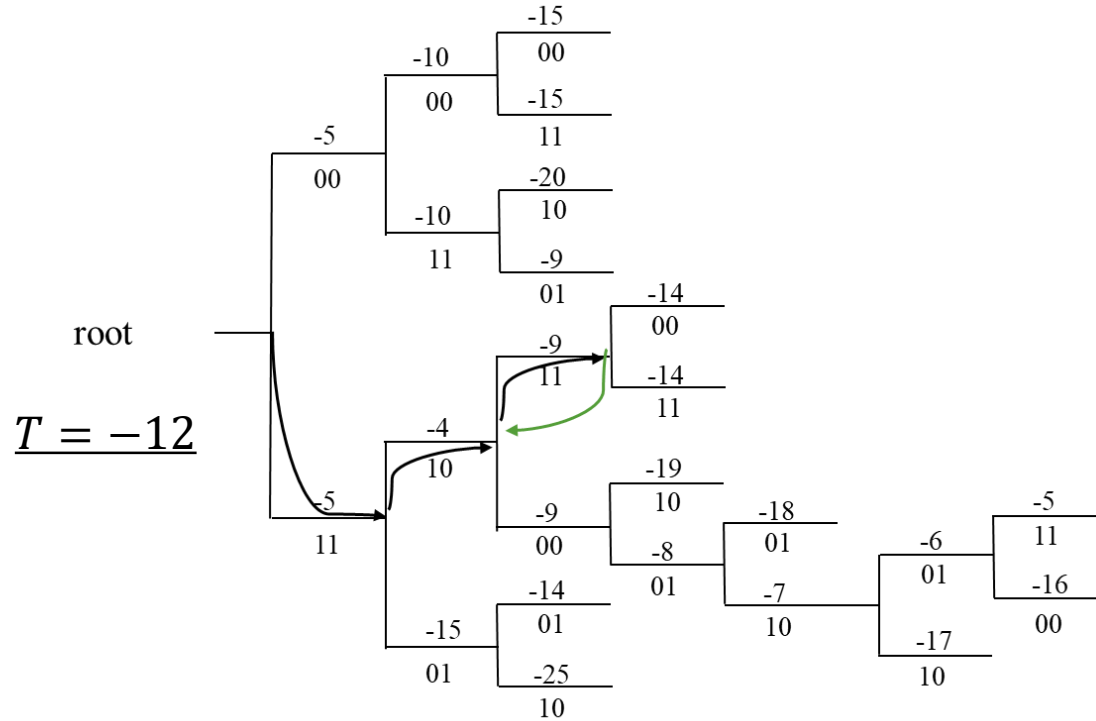
Exploration of the framed (by green dash) paths cannot reach the leaf.





§ 5.7* Sequential Decoding

Exploration of the remaining paths over the tree.



$T = -12$

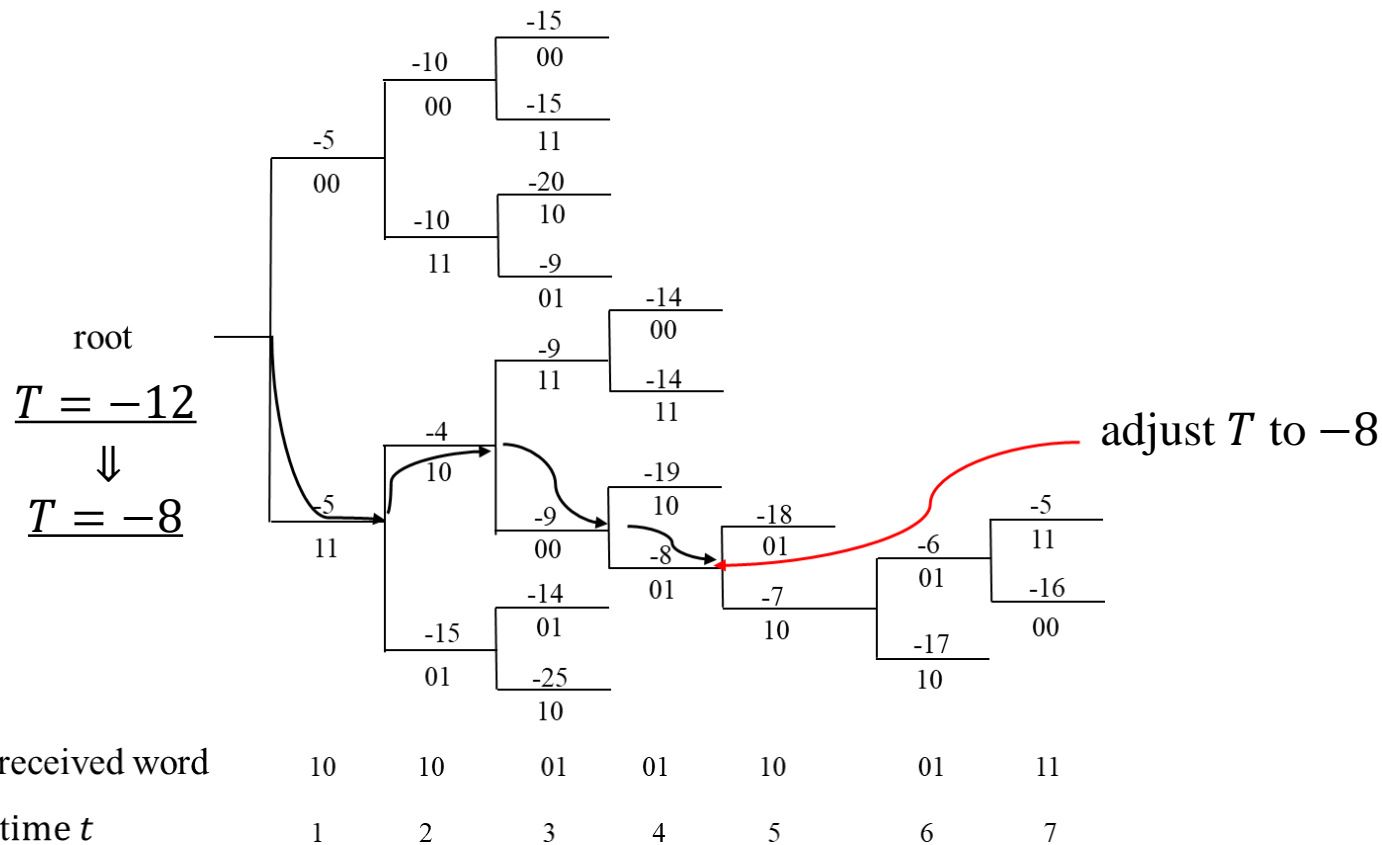
received word	10	10	01	01	10	01	11
time t	1	2	3	4	5	6	7



§ 5.7* Sequential Decoding

If reach a node for the first time and its path metric

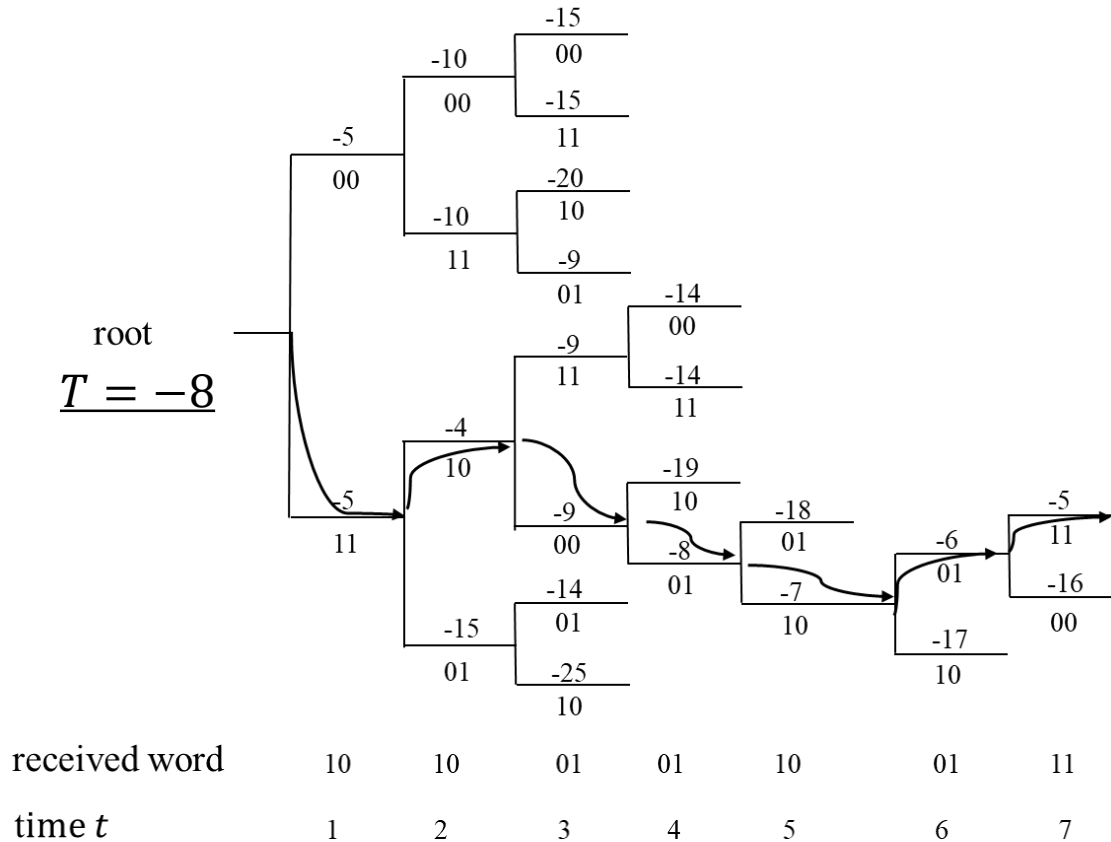
$$\Lambda = -8 \geq T + \Delta = -8, \text{ let } T + \Delta \rightarrow T, \text{ s.t. } \Lambda - \Delta < T \leq \Lambda.$$





§ 5.7* Sequential Decoding

When decoder reach the leaf node, the decoding is completed and the path $[\hat{m}_1 \hat{m}_2 \hat{m}_3 \hat{m}_4 \hat{m}_5 \hat{m}_6 \hat{m}_7] = [1 0 1 1 1 0 0]$ is output.





§ 5.7* Sequential Decoding

- Parameter selecting:
 - With a small Δ , the search is precise, but it often falls back to the root node to adjust the threshold T .
 - With a large Δ , the searching speed may be fast. But it is easy to explore the wrong path and stuck, causing an increase in complexity.
 - Choose Δ empirically, such as 2 or 4.
- Fano path metric Λ is used for comparing paths of different lengths. The path with the maximum Fano path metric is considered the most reliable one.
- If the path is correct, Λ would increase slightly. If current path deviates from the correct path, Λ would continue to decline when going forward.
- Fano decoder will linger between several paths with similar path metrics, which is the main cause of its latency and complexity. In communication systems, the uncertainty of the duration from the beginning to the end of decoding is often undesirable.